Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 198 (2004) 469–499
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A front-tracking/front-capturing method for the simulation of 3D multi-fluid flows with free surfaces

F.S. de Sousa [a,*], N. Mangiavacchi [b], L.G. Nonato [a], A. Castelo [a], M.F. Tomé [a], V.G. Ferreira [a], J.A. Cuminato [a], S. McKee [c]

[a] *Departamento de Ciências de Computação e Estatística, Instituto de Ciências Matemáticas e de Computação de São Carlos – USP, Cx.P. 668, 13560-970, São Carlos, SP, Brazil*
[b] *Departamento de Engenharia Mecânica, Universidade do Estado do Rio de Janeiro, Rua São Francisco Xavier, 524, 20550-900, Rio de Janeiro, RJ, Brazil*
[c] *Department of Mathematics, University of Strathclyde, G11XH Glasgow, Scotland, UK*

Received 22 July 2003; received in revised form 20 January 2004; accepted 23 January 2004
Available online 27 February 2004

Abstract

A method for simulating incompressible, imiscible, unsteady, Newtonian, multi-fluid flows with free surfaces is described. A sharp interface separates fluids of different density and viscosity. Surface and interfacial tensions are also considered and the required curvature is geometrically approximated at the fronts by a least squares quadratic fitting. To remove small undulations at the fronts, a mass-conserving filter is employed. The numerical method employed to solve the Navier–Stokes equations is based on the GENSMAC-3D front-tracking method. The velocity field is computed using a finite-difference scheme on an Eulerian grid. The free-surface and the interfaces are represented by an unstructured Lagrangian grid moving through an Eulerian grid. The method was validated by comparing the numerical results with analytical results for a number of simple problems. Complex numerical simulations show the capability and emphasize the robustness of this new method.

Keywords: Front-tracking method; Multi-fluid flows; Numerical simulation; Surface tension; Free surface flows; Finite-difference method

1. Introduction

Numerical simulations of fluid flows are relevant to problems found in various industries such as oil, nuclear, chemical and the food industry. Applications involving flows of several fluids, called multi-fluid flows, include heavy oil transport, food processing, co-extrusion and bubble flows. If the fluid involves both gas and liquid, the fluid is referred to as two-phase.

* Corresponding author. Present address: Roland Holstlaan 465, 2624HM Delft, Netherlands.
E-mail address: fsimeoni@lcad.icmc.usp.br (F.S. de Sousa).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2004.01.032

Multi-fluid flows are complicated because of the many physical phenomena that need to be taken into account. Prime among these is surface or interfacial tension between any two fluids. This can be particularly difficult when the two fluids are very different: a simple example is that of bubbles in water, where the density ratio is about 1000 to 1.

There are essentially two techniques for approximating the interface: *front capturing* and *front tracking*. Front-capturing techniques are characterized by treating the interface as a high variation region with no explicit elements to represent the interface. With this approach it is arguably easier to deal with topological changes in the interface (or interfaces) like merging and breaking. However, a major disadvantage of this technique is the interface diffusion over several cells, resulting in loss of precision.

Early work on front capturing goes back to Glimm and his co-workers (see, e.g. [10]) where they represented the moving front by a connected set of points which form a moving internal boundary. An irregular grid is then constructed in order to calculate the evolution of the fluid in the vicinity of the interface. A special finite difference is then employed on the irregular grid. More recently the level set approach [25] has found prominence. This approach employs a smooth function ϕ, called the level set function, to represent the free surface. Liquid regions are regions which have $\phi(\mathbf{x},t) > 0$ while the region containing the other fluid are regions in which $\phi(\mathbf{x},t) < 0$. The free surface is the set of points such that $\phi(\mathbf{x},t) = 0$. The unit normal and the mean curvature are then easily computed. They are given by

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|} \quad \text{and} \quad \kappa = \nabla\cdot\frac{\nabla\phi}{|\nabla\phi|},$$

respectively. This method was introduced by Osher and Sethian [17], and strongly promoted in two books by Sethian [19,20].

Essentially front tracking involves explicit computational elements moving through a (static) Eulerian grid. This approach can be more accurate than the above, but can require additional computational resources. Furthermore, to take into account topological changes in the interface, the implementation can become more complex. A front-tracking method has been described by Tryggvason and his co-workers [7,8,30] in which the interface (or interfaces) are considered to be a linked set of points forming an unstructured mesh. This unstructured mesh moves through a staggered Eulerian grid upon which the fluid velocity and pressure are calculated.

The *Marker-and-Cell* (MAC) method was first introduced by Harlow and Welch [11]. It was specifically designed for calculating incompressible viscous free surface flows. It is a finite-difference method based on a staggered grid that employs the primitive variables of velocity and pressure. Due to computational difficulties, Amsden and Harlow [1] developed a projection method called *Simplified Marker-and-Cell* (SMAC) whereby essentially the velocity and pressure field are calculated sequentially. In the years that followed many people have studied and worked on the SMAC method (e.g. [2,12,15,31,32]). More recently, and motivated by the SMAC philosophy, Tomé and McKee [26] developed the GENSMAC method for two-dimensional incompressible viscous fluids with free surfaces in an arbitrary complex domain. Extensions of the GENSMAC method for axisymmetric [27] and three-dimensional [5,28] flows were also developed.

The starting point for this work is the GENSMAC-3D method [28]. However, rather than dealing with a single fluid, we shall be concerned with an arbitrary number of Newtonian fluids with different densities and viscosities, all separated by interfaces which must be determined. In addition, there is also a free surface or free surfaces that must be treated. The fronts, both free surfaces and interfaces, will lie on Lagrangian meshes that will move through an Eulerian grid on which the fluid velocity and pressure will be calculated. The approach is similar to Esmaeeli and Tryggvason [7,8], but there are significant differences, both in the discretization employed and in the way in which the interfacial forces are calculated. Both the normal to the surface and its curvature are required. However, before these may be calculated it is necessary to remove

sub-cell undulations. Thus the technique for determining the normal and the curvature is two-stage: a novel technique for sub-cell undulations removal is followed by the calculation of the normal and the curvature.

2. Formulation and numerical method

The governing equations for incompressible flows are the Navier–Stokes equations. The approach used here is to consider the density and viscosity as variables over the whole domain, but constant in each specific fluid. Thus, each fluid is taken to be incompressible, and the continuity equation is valid over the whole domain. The pressure is discontinuous along the interfaces, with a discontinuity proportional to the local surface tension and curvature. Hence it behaves like a Heaviside function, and in the Navier–Stokes equations the gradient of this Heaviside function appears. Therefore, the interface can be identified as the region where ∇H is nonzero.

When taking into account the interfacial tension forces and the fact that the density and the viscosity are variable over the whole region, the nondimensional momentum equation is given by

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\frac{1}{\rho}\nabla p + \frac{1}{\rho\, Re}\nabla \cdot \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}}) + \frac{1}{Fr^2}\mathbf{g} - \frac{\sigma_1 \kappa}{\rho\, We}\nabla H. \tag{1}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ denotes the velocity field, $p = p(\mathbf{x}, t)$ is the pressure field, ρ and μ are the density and viscosity of the fluid all in nondimensional form; these are variable in whole domain, but constant and in general different in each phase. Here $\mathbf{g}$ denotes the gravitational field. Furthermore, $Re = \rho_0 LU/\mu_0$, $Fr = U/\sqrt{Lg}$ and $We = \rho_0 LU^2/\sigma_0$ denotes the Reynolds number, the Froude number and the Weber number, respectively. Here, L and U are the length and velocity scales, ρ_0, μ_0 and σ_0 are the reference values of density, viscosity and surface tension, and g denotes the gravitational constant.

All the fluids are assumed to be incompressible, so the velocity field is divergence free, that is

$$\nabla \cdot \mathbf{u} = 0. \tag{2}$$

Eqs. (1) and (2) are solved based on the GENSMAC-3D method [28], using an extension of the methodology described in [18] for the two-dimensional case. It is supposed that the velocity field $\mathbf{u}(\mathbf{x}, t_0)$ is known at a given time t_0, and the boundary conditions for the velocity and pressure are given. The updated velocity field $\mathbf{u}(\mathbf{x}, t)$, at $t = t_0 + \Delta t$, is calculated using the following algorithm:

(1) Let $\tilde{p}(\mathbf{x}, t)$ be a pressure field which satisfies the correct pressure conditions on the free surface. This pressure field is computed according to the required boundary stress conditions.
(2) The intermediate velocity field $\tilde{\mathbf{u}}(\mathbf{x}, t)$ is computed by the explicitly discretized form of the momentum equations

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\frac{1}{\rho}\nabla \tilde{p} + \frac{1}{\rho\, Re}\nabla \cdot \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}}) + \frac{1}{Fr^2}\mathbf{g} - \frac{\sigma_1 \kappa}{\rho\, We}\nabla H \tag{3}$$

with $\tilde{\mathbf{u}}(\mathbf{x}, t_0) = \mathbf{u}(\mathbf{x}, t_0)$ using the correct boundary conditions for $\mathbf{u}(\mathbf{x}, t_0)$. It can be shown [26] that $\tilde{\mathbf{u}}(\mathbf{x}, t)$ possesses the correct vorticity at time t. However, $\tilde{\mathbf{u}}(\mathbf{x}, t)$ does not satisfy (2). Let

$$\mathbf{u}(\mathbf{x}, t) = \tilde{\mathbf{u}}(\mathbf{x}, t) - \frac{1}{\rho}\nabla \psi(\mathbf{x}, t) \tag{4}$$

with

$$\nabla \cdot \frac{1}{\rho}\nabla \psi(\mathbf{x}, t) = \nabla \cdot \tilde{\mathbf{u}}(\mathbf{x}, t). \tag{5}$$

Thus, $\mathbf{u}(\mathbf{x},t)$ now satisfies (2) and the vorticity remains unchanged. Therefore, $\mathbf{u}(\mathbf{x},t)$ is identified as the updated velocity field at time t.

(3) Solve the elliptic equation (5).
(4) Compute the velocity from (4).
(5) Compute the pressure using

$$p(\mathbf{x},t) = \tilde{p}(\mathbf{x},t) + \frac{\psi(\mathbf{x},t)}{\Delta t}. \tag{6}$$

(6) Update the positions of the marker particles.

The last step in the calculation involves moving the marker particles to their new positions. These are virtual particles whose coordinates are stored and updated at the end of each cycle by solving $\mathrm{d}\mathbf{x}/\mathrm{d}t = \mathbf{u}$ by Euler's method. This provides the new coordinates of every particle, allowing us to determine whether or not it has moved to a new computational cell or if it has left the containment region through an outlet. Using the front-tracking methodology [28,30], only marker particles on the free surface and the interfaces need to be considered.

For the solution of (3), appropriate boundary conditions are required. At solid walls null velocities are enforced. As implied by the title of this work, the proposed method is capable of dealing with low density fluids in two different ways: as one of the phases of the flow or as an inert phase, thus introducing a moving free boundary. At the free surface, the boundary conditions for pressure and velocity, assuming zero viscous stress in the empty phase, are given by

$$(\mathbf{S}\cdot\mathbf{n})\cdot\mathbf{n} = p_{\mathrm{cap}}, \quad (\mathbf{S}\cdot\mathbf{n})\cdot\mathbf{m}_1 = 0, \quad (\mathbf{S}\cdot\mathbf{n})\cdot\mathbf{m}_2 = 0, \tag{7}$$

where $\mathbf{n}$, $\mathbf{m}_1$ and $\mathbf{m}_2$ are the local, normal and tangential vectors to the free surface. $\mathbf{S}$ is the viscous stress tensor, given by

$$\mathbf{S} = -p\mathbf{I} + \frac{\mu}{Re}\left[(\nabla\mathbf{u}) + (\nabla\mathbf{u})^{\mathrm{T}}\right], \tag{8}$$

and $p_{\mathrm{cap}} = \sigma\kappa/We$ is the capillary pressure, originating from the effects of the surface tension σ. The elliptic equation (5) is solved using the conjugate gradient method, satisfying homogeneous Dirichlet boundary conditions at the free surface and homogeneous Neumann boundary conditions at the solid boundaries. However, as the density variation across the interface increases, more iterations are required for the convergence of the method. In these cases, a diagonal preconditioner was used to speed-up the convergence of the conjugate gradient method.

3. Discretizations

In a similar manner to MAC [33], SMAC [1] and GENSMAC [26] methods, Eqs. (3)–(5) are discretized by finite differences on a staggered grid. Fig. 1 shows an example of a staggered grid cell and the position of the variables in this cell: the velocities are calculated on the faces, and the other variables (pressure, density and viscosity) are computed at the cell center.

Consider the x component of the nondimensional momentum equation, expressed in Cartesian coordinates

$$\begin{aligned}\frac{\partial u}{\partial t} + \frac{\partial(u^2)}{\partial x} + \frac{\partial(uv)}{\partial y} + \frac{\partial(uw)}{\partial z} = &-\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\mu}{\rho Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + \frac{1}{\rho Re}\left[2\frac{\partial\mu}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial\mu}{\partial y}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)\right.\\ &\left. + \frac{\partial\mu}{\partial z}\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right] + \frac{1}{Fr^2}g_x - \frac{\sigma_1\kappa}{\rho We}\frac{\partial H}{\partial x}.\end{aligned} \tag{9}$$

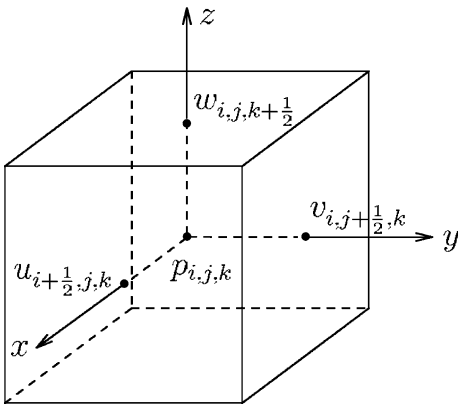

Fig. 1. Position of the variables in a cell. The velocities are computed on the faces of the cell, and the pressure (p), density (ρ), viscosity (μ) evaluated at the cell center.

This equation can be written in the compact form

$$\frac{\partial u}{\partial t} + C_x = -\frac{1}{\rho}\frac{\partial p}{\partial x} + V_x^1 + V_x^2 + \frac{1}{Fr^2}g_x - \frac{\sigma_I \kappa}{\rho\, We}\frac{\partial H}{\partial x}, \tag{10}$$

where

$$C_x = \frac{\partial(u^2)}{\partial x} + \frac{\partial(uv)}{\partial y} + \frac{\partial(uw)}{\partial z}, \tag{11}$$

$$V_x^1 = \frac{\mu}{\rho\, Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right), \tag{12}$$

$$V_x^2 = \frac{1}{\rho\, Re}\left[2\frac{\partial \mu}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial \mu}{\partial y}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) + \frac{\partial \mu}{\partial z}\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right] \tag{13}$$

are the convective (C_x) and viscous (V_x^1, V_x^2) terms of (9). Eq. (10) is discretized by finite differences at the point $(i+\frac{1}{2}, j, k)$ of the cell as follows. The time derivative is discretized by the first-order forward difference. All the other terms of (10) are discretized at the time level n, i.e. the discretization is considered to be explicit. Thus, in the following equations, the index n is omitted to simplify the notation. The pressure gradient is discretized by central differences. Notice that it is necessary to evaluate $\rho_{i+\frac{1}{2},j,k}$ at the face of the cell, and we compute an approximation by averaging the known neighboring values. Thus,

$$\frac{1}{\rho}\frac{\partial p}{\partial x}\bigg|_{i+\frac{1}{2},j,k} = \frac{2(p_{i+1,j,k} - p_{i,j,k})}{\Delta x(\rho_{i+1,j,k} + \rho_{i,j,k})}. \tag{14}$$

To compute the viscosity, we use the *harmonic mean*

$$\mu_{i+\frac{1}{2},j,k} = 2\left(\frac{1}{\mu_{i+1,j,k}} + \frac{1}{\mu_{i,j,k}}\right)^{-1}, \tag{15}$$

this is more accurate than simple averaging [29]. Thus, the viscous term V_x^1 using (15) becomes

$$V_x^1\big|_{i+\frac{1}{2},j,k} = \frac{4}{Re\left(\frac{1}{\mu_{i+1,j,k}}+\frac{1}{\mu_{i,j,k}}\right)\left(\rho_{i+1,j,k}+\rho_{i,j,k}\right)} \cdot \left[\frac{u_{i-\frac{1}{2},j,k} - 2u_{i+\frac{1}{2},j,k} + u_{i+\frac{3}{2},j,k}}{\Delta x^2} + \frac{u_{i+\frac{1}{2},j-1,k} - 2u_{i+\frac{1}{2},j,k} + u_{i+\frac{1}{2},j+1,k}}{\Delta y^2} + \frac{u_{i+\frac{1}{2},j,k-1} - 2u_{i+\frac{1}{2},j,k} + u_{i+\frac{1}{2},j,k+1}}{\Delta z^2}\right]. \tag{16}$$

The values of viscosity on the edges of the cell are computed by an analogous operation to (15). Finally, the discretization of V_x^2 becomes

$$\begin{aligned} V_x^2\big|_{i+\frac{1}{2},j,k} = \frac{1}{Re(\rho_{i+1,j,k}+\rho_{i,j,k})} \Bigg\{ & \frac{\left(\mu_{i+1,j,k}-\mu_{i,j,k}\right)\left(u_{i+\frac{3}{2},j,k}-u_{i-\frac{1}{2},j,k}\right)}{\Delta x^2} \\ & + \frac{4}{\Delta y}\left[\left(\frac{1}{\mu_{i,j,k}}+\frac{1}{\mu_{i+1,j,k}}+\frac{1}{\mu_{i,j+1,k}}+\frac{1}{\mu_{i+1,j+1,k}}\right)^{-1} - \left(\frac{1}{\mu_{i,j,k}}+\frac{1}{\mu_{i+1,j,k}}+\frac{1}{\mu_{i,j-1,k}}+\frac{1}{\mu_{i+1,j-1,k}}\right)^{-1}\right] \\ & \times \left(\frac{v_{i+1,j+\frac{1}{2},k}+v_{i+1,j-\frac{1}{2},k}-v_{i,j+\frac{1}{2},k}-v_{i,j-\frac{1}{2},k}}{2\Delta x}+\frac{u_{i+\frac{1}{2},j+1,k}-u_{i+\frac{1}{2},j-1,k}}{2\Delta y}\right) \\ & + \frac{4}{\Delta z}\left[\left(\frac{1}{\mu_{i,j,k}}+\frac{1}{\mu_{i+1,j,k}}+\frac{1}{\mu_{i,j,k+1}}+\frac{1}{\mu_{i+1,j,k+1}}\right)^{-1} - \left(\frac{1}{\mu_{i,j,k}}+\frac{1}{\mu_{i+1,j,k}}+\frac{1}{\mu_{i,j,k-1}}+\frac{1}{\mu_{i+1,j,k-1}}\right)^{-1}\right] \\ & \times \left(\frac{w_{i+1,j,k+\frac{1}{2}}+w_{i+1,j,k-\frac{1}{2}}-w_{i,j,k+\frac{1}{2}}-w_{i,j,k-\frac{1}{2}}}{2\Delta x}+\frac{u_{i+\frac{1}{2},j,k+1}-u_{i+\frac{1}{2},j,k-1}}{2\Delta z}\right)\Bigg\}. \end{aligned}$$

The discretization of the momentum equations in the y and z directions is obtained in a similar manner. The nonlinear terms in the momentum equation are discretized using the high order upwind scheme VONOS, omitted here for brevity. Full details may be found in Ferreira et al. [9]. This scheme is accurate and stable for simulations where the Reynolds number is high.

Using the tracking particles, the free surface and interfaces are approximated by a piecewise linear surface and computationally represented by the "half-edge" data structure [14].

The flow properties are evaluated in a three-dimensional uniform grid, in which every cell, at each time step, is classified according to its position relative to the fluids and the rigid boundaries. Cells with more than half of their volume in a container are classified as BOUNDARY (**B**) cells; the same criteria are used for the INFLOW (**I**) cells. Any cell completely inside the fluid is classified as a FULL (**F**) cell, those completely outside the fluid are EMPTY (**E**) cells. Those containing marker particles are SURFACE (**S**) cells if they have at least one EMPTY neighbor. This criteria is applied to each fluid in the simulation, and the INTERFACE cells are identified as the cells that are SURFACE or FULL for more than one fluid at the same time. Fig. 2 shows an example of cell flagging in a two-dimensional case, with a classification for each fluid.

The cell classification is updated at each time step using the information provided by the Lagrangian mesh constituted by the tracking particles. Since time advancement is restricted by a CFL type condition, the interfaces move at most one cell at each time step. Hence, it is only necessary to update cell classifications in the neighborhood of the surface cells. The procedure can be described in three steps. First, the cells that contain particles are identified. If any identified cell is EMPTY then it is marked as SURFACE. Second, cells marked as SURFACE and that do not contain particles anymore are reflagged either as

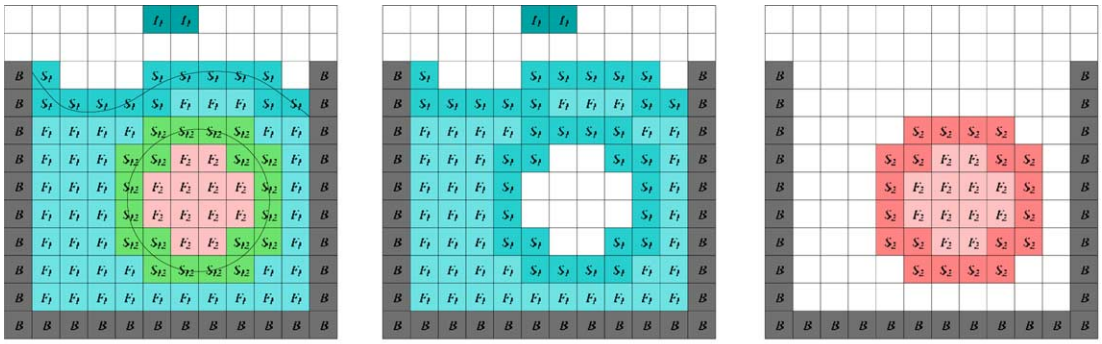

Fig. 2. Multi-fluid cell flagging example: (a) classification for fluids 1 and 2; (b) classification only for fluid 1; (c) classification only for fluid 2.

EMPTY or FULL according to the neighboring flags. Third, the FULL cells that contain particles and have any EMPTY neighboring cell are marked as SURFACE. This approach allows us to take surface merging into account. However, more complex topological changes such as those involving the splitting of surfaces could be implemented applying a strategy similar to Shin and Juric [21].

In order to apply the free surface boundary conditions (7) in each **S** cell, approximations for the surface normals were needed. These have been obtained according to the classification of the neighboring cells [28]. For example: $\mathbf{n}_C = (\pm 1, 0, 0)$, $\mathbf{n}_C = (0, \pm 1, 0)$ or $\mathbf{n}_C = (0, 0, \pm 1)$ if only one neighbor is an **E** cell; $\mathbf{n}_C = (\pm\frac{\sqrt{2}}{2}, \pm\frac{\sqrt{2}}{2}, 0)$, $\mathbf{n}_C = (\pm\frac{\sqrt{2}}{2}, 0, \pm\frac{\sqrt{2}}{2})$ or $\mathbf{n}_C = (0, \pm\frac{\sqrt{2}}{2}, \pm\frac{\sqrt{2}}{2})$ if there are two neighboring **E** cells, and; $\mathbf{n}_C = (\pm\frac{\sqrt{3}}{3}, \pm\frac{\sqrt{3}}{3}, \pm\frac{\sqrt{3}}{3})$ if there are three neighboring **E** cells in the x, y and z directions. Note the notation $\mathbf{n}_C$ here means the normal at the center of the **S** cell being considered. In order to implement surface tension effects it is also necessary to estimate the surface curvature at the center of each surface cell, and to take into account sub-cell surface tension effects. In the following sections the methodology employed in the implementation of the surface tension effects is described. This methodology results in a better estimate of the surface normal which is then used in the computation of the capillary pressure.

3.1. Interfacial tension effects

To implement the interfacial tension effects into the momentum equation, it is necessary to determine the resulting forces that need to be applied in the discretized domain.

If the mesh is chosen to be sufficiently fine, then the interface can be approximated by the cells containing the front. This approach results in an interface that is at least one cell thick (in most points on the surface the interface will be only one cell thick). So, we have to determine the interfacial tension forces in those cells by a discretized form of

$$\mathbf{f} = -\frac{\sigma_I \kappa}{\rho\, We} \nabla H,$$

where H is a Heaviside function that is 1 inside the particular fluid under consideration and 0 outside.

It is convenient to define one Heaviside function H_λ for each fluid λ in the discretized domain taking into account the region defined by the interfacial cells. The Heaviside function is constructed at each time step according to the cell classification, and is given by

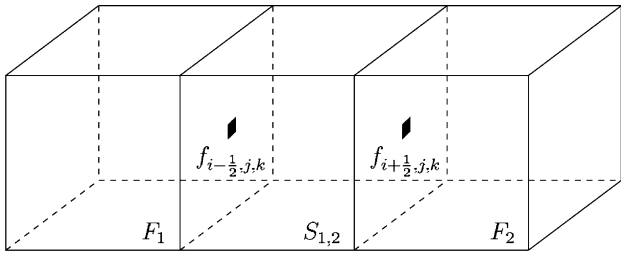

Fig. 3. Interface cell between fluids 1 and 2, with full cells in the opposite faces.

$$H_\lambda(x,y,z) = \begin{cases} 1 & \text{if } (x,y,z) \text{ is inside a FULL cell or is} \\ & \text{inside an INTERFACE cell of the fluid } \lambda, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

such that the force to be applied in each cell is given by

$$\mathbf{f} = -\frac{\sigma_{\mathrm{I}}}{\rho\, We}\frac{1}{n}\sum_{\lambda=1}^{n}\kappa_\lambda \nabla H_\lambda, \tag{18}$$

where n is the number of fluids involved. Each fluid supplies a different curvature and Heaviside function. For each fluid λ the force is applied at each face of the interfacial cell, that is in contact with an empty cell. For example, to calculate the force contribution in a single face $(i-\frac{1}{2},j,k)$, as shown in Fig. 3, we have two Heaviside functions to be discretized, H_1 and H_2.

$$\left.\frac{\partial H_1}{\partial x}\right|_{i-\frac{1}{2},j,k} = \frac{H_1(i\Delta x, j\Delta y, k\Delta z) - H_1((i-1)\Delta x, j\Delta y, k\Delta z)}{\Delta x} = 0,$$

$$\left.\frac{\partial H_2}{\partial x}\right|_{i-\frac{1}{2},j,k} = \frac{H_2(i\Delta x, j\Delta y, k\Delta z) - H_2((i-1)\Delta x, j\Delta y, k\Delta z)}{\Delta x} = \frac{1}{\Delta x},$$

and then, the resulting force to be applied to face $(i-\frac{1}{2},j,k)$ is given by

$$f_{i-\frac{1}{2},j,k} = \frac{-\sigma_{\mathrm{I}}}{\rho_{i-\frac{1}{2},j,k}We}\frac{1}{2}\left(\kappa_1\left.\frac{\partial H_1}{\partial x}\right|_{i-\frac{1}{2},j,k} + \kappa_2\left.\frac{\partial H_2}{\partial x}\right|_{i-\frac{1}{2},j,k}\right) = \frac{-\sigma_{\mathrm{I}}\kappa_2}{2\Delta x\rho_{i-\frac{1}{2},j,k}We}.$$

The other cases are obtained following this procedure. The gradient of the Heaviside functions gives the correct direction of action of the applied forces in any configuration.

4. Curvature calculation

Surface and interfacial tension effects are incorporated using an approach similar to the one described in [4] for the two-dimensional case. The computation of the surface and interfacial tension is performed at two levels: first at the sub-grid level, where small undulations in the fronts (free surface and interfaces) are eliminated, and second at cell level, where the curvature at each surface or interface cell is approximated. This approximation will be used in the implementation of the pressure boundary condition at the free surface and in the calculation of the interfacial forces described above.

The curvature is approximated by the surface that best fits the points (the position of the marker particles) in that cell and its neighbors. The results presented in [22] suggest the use of an approximation by the paraboloid

$$\pi(x,y) = ax^2 + bxy + cy^2 + dx + ey + f. \tag{19}$$

In this approximation, we first need to determine the normal vector at the cell center. We can summarize the method of curvature approximation by the following steps:

(1) Given a surface or interface cell, consider all points that lie inside a sphere S_δ with its center at the cell center with a given radius δ.
(2) Fit a plane to these points, and compute its normal vector.
(3) Take a new coordinate system normal to the surface, using the previous computed normal vector.
(4) Consider a new sphere centered at the cell center, S_ε with a chosen radius ε, and then map the points in this sphere to the new coordinate system.
(5) Fit the surface given by (19) to the mapped points, using the new coordinate system.
(6) Compute the total curvature of the fitted surface using

$$\kappa = \frac{\frac{\partial^2 \pi}{\partial \xi^2}}{\left[1 + \left(\frac{\partial \pi}{\partial \xi}\right)^2\right]^{\frac{3}{2}}} + \frac{\frac{\partial^2 \pi}{\partial \eta^2}}{\left[1 + \left(\frac{\partial \pi}{\partial \eta}\right)^2\right]^{\frac{3}{2}}}, \tag{20}$$

where π is the fitted surface satisfying (19), and (ξ, η, ζ) are the coordinates of the new coordinate system. Note the ζ-axis is parallel to the normal vector.

In this method, the quality of the approximations is directly related to the chosen radii δ and ε. Typically δ is chosen to be of the order of the cell size and ε 1.5 times larger. For more detailed discussion of the influence of these constants see [22], where the optimum choice for these numbers is discussed.

How the normal vector and the curvature approximations are made will be described in detail in the next sub-sections.

4.1. Normal vector approximation

The region S_δ associated with each cell $\mathbf{S}$ will include a number of points. Let us call that number m; so $m = m(\delta)$. Let $\mathbf{x}_i = (x_i, y_i, z_i)$, $i = 1, \ldots, m_\delta$, be the number of points inside S_δ and thus the number of points in cell $\mathbf{S}$ and its surrounding neighbors. The equation of the plane will be either of $z = f(x,y)$, $y = f(x,z)$ or $x = f(y,z)$, according to the maximum absolute value of cell normal vector components. For example, if $|\langle \mathbf{n}_\mathrm{C}, (0,0,1)\rangle| > |\langle \mathbf{n}_\mathrm{C}, (1,0,0)\rangle|$ and $|\langle \mathbf{n}_\mathrm{C}, (0,0,1)\rangle| > |\langle \mathbf{n}_\mathrm{C}, (0,1,0)\rangle|$, where $\langle , \rangle$ denotes the inner product, the equation of the plane is given by $z = f(x,y) = ax + by + c$, and we need to determine the coefficients a, b and c, such that the plane $z = ax + by + c$ approximates the given points. The least squares approximation can be obtained by solving the normal equations

$$\begin{pmatrix} \langle \mathbf{a}, \mathbf{a}\rangle & \langle \mathbf{a}, \mathbf{b}\rangle & \langle \mathbf{a}, \mathbf{c}\rangle \\ \langle \mathbf{b}, \mathbf{a}\rangle & \langle \mathbf{b}, \mathbf{b}\rangle & \langle \mathbf{b}, \mathbf{c}\rangle \\ \langle \mathbf{c}, \mathbf{a}\rangle & \langle \mathbf{c}, \mathbf{b}\rangle & \langle \mathbf{c}, \mathbf{c}\rangle \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \langle \mathbf{z}, \mathbf{a}\rangle \\ \langle \mathbf{z}, \mathbf{b}\rangle \\ \langle \mathbf{z}, \mathbf{c}\rangle \end{pmatrix}, \tag{21}$$

where $\mathbf{a} = (x_1, \ldots, x_m)^\mathrm{T}$, $\mathbf{b} = (y_1, \ldots, y_m)^\mathrm{T}$, $\mathbf{c} = (1, \ldots, 1)^\mathrm{T}$ and $\mathbf{z} = (z_1, \ldots, z_m)^\mathrm{T}$, where by m we mean m_δ. In this case, the normal vector to the plane $z = ax + by + c$ is given by

$$\mathbf{n}_\mathrm{S} = \frac{(a, b, -1)}{\|(a, b, -1)\|}. \tag{22}$$

For the other cases, this process is analogous. This procedure determines $\mathbf{n}_S$, but the sign may be incorrect. The correct sign can be determined by comparing $\mathbf{n}_S$ with the normal at the cell center $\mathbf{n}_C$, previously obtained (see Section 3). If $\langle \mathbf{n}_C, \mathbf{n}_S \rangle > 0$ the sign is correct; otherwise the sign must be reversed.

4.2. Quadratic fitting

Now let $\mathbf{x}_i = (x_i, y_i, z_i)$, $i = 0, \ldots, m_\varepsilon$, be the points lying in S_ε. With the normal vector $\mathbf{n}_S$ given by the plane fitting technique of Section 4.1, we choose another two tangent vectors at the surface such that they, together with the normal vector, form an orthonormal basis. First, we choose two of the three canonical vectors $\mathbf{e}_1 = (1,0,0)$, $\mathbf{e}_2 = (0,1,0)$ and $\mathbf{e}_3 = (0,0,1)$ such that their projections on the direction of the $\mathbf{n}_S$ vector are smallest. Second, we apply the Gramm Schmidt method to obtain the orthonormal basis. Thus, we have a new coordinate system and the points $\mathbf{x}_i$ can be written using this new coordinate system as $\boldsymbol{\xi}_i = (\xi_i, \eta_i, \zeta_i)$. Then, we can fit a paraboloid given by $\hat{\pi}(\xi, \eta) = a\xi^2 + b\xi\eta + c\eta^2 + d\xi + e\eta + f$. The parameters a, b, c, d, e, and f are obtained by solving

$$\begin{pmatrix} \langle \mathbf{a},\mathbf{a}\rangle & \langle \mathbf{a},\mathbf{b}\rangle & \langle \mathbf{a},\mathbf{c}\rangle & \langle \mathbf{a},\mathbf{d}\rangle & \langle \mathbf{a},\mathbf{e}\rangle & \langle \mathbf{a},\mathbf{f}\rangle \\ \langle \mathbf{b},\mathbf{a}\rangle & \langle \mathbf{b},\mathbf{b}\rangle & \langle \mathbf{b},\mathbf{c}\rangle & \langle \mathbf{b},\mathbf{d}\rangle & \langle \mathbf{b},\mathbf{e}\rangle & \langle \mathbf{b},\mathbf{f}\rangle \\ \langle \mathbf{c},\mathbf{a}\rangle & \langle \mathbf{c},\mathbf{b}\rangle & \langle \mathbf{c},\mathbf{c}\rangle & \langle \mathbf{c},\mathbf{d}\rangle & \langle \mathbf{c},\mathbf{e}\rangle & \langle \mathbf{c},\mathbf{f}\rangle \\ \langle \mathbf{d},\mathbf{a}\rangle & \langle \mathbf{d},\mathbf{b}\rangle & \langle \mathbf{d},\mathbf{c}\rangle & \langle \mathbf{d},\mathbf{d}\rangle & \langle \mathbf{d},\mathbf{e}\rangle & \langle \mathbf{d},\mathbf{f}\rangle \\ \langle \mathbf{e},\mathbf{a}\rangle & \langle \mathbf{e},\mathbf{b}\rangle & \langle \mathbf{e},\mathbf{c}\rangle & \langle \mathbf{e},\mathbf{d}\rangle & \langle \mathbf{e},\mathbf{e}\rangle & \langle \mathbf{e},\mathbf{f}\rangle \\ \langle \mathbf{f},\mathbf{a}\rangle & \langle \mathbf{f},\mathbf{b}\rangle & \langle \mathbf{f},\mathbf{c}\rangle & \langle \mathbf{f},\mathbf{d}\rangle & \langle \mathbf{f},\mathbf{e}\rangle & \langle \mathbf{f},\mathbf{f}\rangle \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = \begin{pmatrix} \langle \mathbf{z},\mathbf{a}\rangle \\ \langle \mathbf{z},\mathbf{b}\rangle \\ \langle \mathbf{z},\mathbf{c}\rangle \\ \langle \mathbf{z},\mathbf{d}\rangle \\ \langle \mathbf{z},\mathbf{e}\rangle \\ \langle \mathbf{z},\mathbf{f}\rangle \end{pmatrix}, \tag{23}$$

where $\mathbf{a} = (\xi_1^2, \ldots, \xi_m^2)^T$, $\mathbf{b} = (\xi_1\eta_1, \ldots, \xi_m\eta_m)^T$, $\mathbf{c} = (\eta_1^2, \ldots, \eta_m^2)^T$, $\mathbf{d} = (\xi_1, \ldots, \xi_m)^T$, $\mathbf{e} = (\eta_1, \ldots, \eta_m)^T$, $\mathbf{f} = (1, \ldots, 1)^T$, and $\mathbf{z} = (\zeta_1, \ldots, \zeta_m)^T$, where by m we mean m_ε.

The total curvature is then given by

$$\kappa = -2\left(\frac{a}{(1+d^2)^{\frac{3}{2}}} + \frac{c}{(1+e^2)^{\frac{3}{2}}} \right). \tag{24}$$

5. Undulations removal

In many applications, in particular when the Reynolds number is high (larger than 50), small undulations may appear at the free surface and at the interfaces. This is due to variations in the velocity field from cell to cell and may be amplified in regions where the surface area is shrinking. These undulations are frequently much smaller than a cell size and usually they are not present in real flows. They are physically removed by a combination of surface or interfacial tension and viscous effects. A numerical implementation that acts at cell level, like the method presented above, cannot take into account these sub-cell undulations and correctly suppress them.

There are several approaches that can be used to suppress these unphysical undulations such as applying a Gaussian filter. However, in fluid flow simulations it is important that the technique applied does not alter the mass of the flow (or the volume in the case of an incompressible fluid).

We developed a filter called TSUR-3D (TSUR: Trapezoidal Sub-grid Undulations Removal) which is based on the TSUR filter presented in [4] for the two-dimensional case. The TSUR-3D is applied to the unstructured grid, represented by a "*half-edge*" data structure [14]. Fig. 4 shows a typical vertex $\mathbf{v}$, and its corresponding star, formed by the set of vertices $\mathbf{x}_i$ which are connected to it with typical edges $(\mathbf{v}, \mathbf{x}_i)$ and $(\mathbf{x}_i, \mathbf{x}_{i+1})$, and the typical faces $(\mathbf{x}_i, \mathbf{v}, \mathbf{x}_{i+1})$, $i = 0, 1, \ldots, n$, where $\mathbf{x}_{n+1}$ is to be interpreted as $\mathbf{x}_0$.

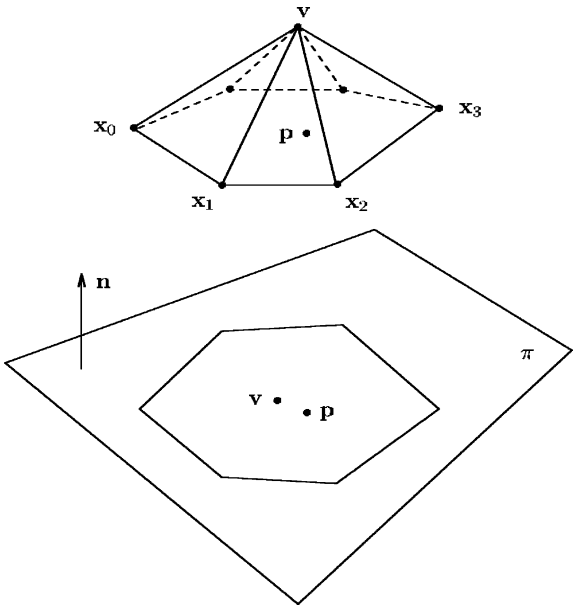

Fig. 4. A typical vertex and its star projected onto the plane π.

Since we are interested in local changes in volume, we define a local volume as the volume bounded by the faces $(\mathbf{x}_i, \mathbf{v}, \mathbf{x}_{i+1})$ and $(\mathbf{x}_i, \mathbf{p}, \mathbf{x}_{i+1})$, $i = 0, 1, \ldots, n$, where $\mathbf{p}$ is an arbitrary point.

The balance procedure first determines a plane π defined by the normal computed from the average of the normals of the vertices $\mathbf{x}_i$, $i = 0, 1, \ldots, n$. The vertex $\mathbf{v}$ and the vertices $\mathbf{x}_i$, $i = 0, 1, \ldots, n$, are projected onto that plane and $\mathbf{v}$ is centered in that plane, and the correct height of the new vertex in the direction of the normal of the plane is chosen so that the local volume is preserved (see Fig. 4).

This procedure smooths small undulations and keeps the volume of the fluid unchanged. The steps of the method are explained in the following sections.

5.1. Normal vector calculation

Both vertex balance and undulation removal procedures require a normal direction, which is obtained using information from the neighboring vertices $\mathbf{x}_i$, $i = 0, 1, \ldots, n$. Let $\mathbf{v}$ be a vertex of the interface, and let $S(\mathbf{v})$ be the star of $\mathbf{v}$. Let $\mathbf{p}$ be the average of the vertices in $S(\mathbf{v})$, given by

$$\mathbf{p} = \frac{1}{n}\sum_{i=0}^{n} \mathbf{x}_i, \tag{25}$$

where $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n \in S(\mathbf{v})$. Thus, we can compute the normal vector by the average of the normal vectors at the faces $(\mathbf{x}_i, \mathbf{p}, \mathbf{x}_{i+1})$ $(i = 0, 1, \ldots, n)$, given by

$$\mathbf{n} = \frac{\sum_{i=0}^{n-1}(\mathbf{x}_i - \mathbf{p}) \times (\mathbf{x}_{i+1} - \mathbf{p}) + (\mathbf{x}_n - \mathbf{p}) \times (\mathbf{x}_0 - \mathbf{p})}{\| \sum_{i=0}^{n-1}(\mathbf{x}_i - \mathbf{p}) \times (\mathbf{x}_{i+1} - \mathbf{p}) + (\mathbf{x}_n - \mathbf{p}) \times (\mathbf{x}_0 - \mathbf{p})\|}, \tag{26}$$

where × denotes the vector product. Note this is a weighted average: normals of faces with greater area will be given greater weight. This equation gives a robust normal direction for this method.

5.2. Vertex balance procedure

The vertex balance procedure is applied at each vertex of the fluid in order to move it to a new position on a line passing through **p**, given by (25) with the direction **n** given by (26), such that the local volume is preserved. Let π be a plane that contains **p**, with normal vector **n**. If **p** is inside the projection of $S(\mathbf{v})$ onto π, we compute the new position of **v** by

$$\mathbf{v}_{\text{new}} = \mathbf{p} + h\mathbf{n}, \tag{27}$$

where h is determined such that the local volume is preserved. This is guaranteed if

$$V = hV_1 \quad \text{or} \quad h = \frac{V}{V_1}, \tag{28}$$

where V is the volume of the polyhedron $(\mathbf{v}, \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{p})$ and V_1 is the volume of the unitary polyhedron $(\mathbf{p} + \mathbf{n}, \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{p})$. This is because the volume of the polyhedron $(\mathbf{p} + h\mathbf{n}, \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{p})$ is equal to h times the volume of the unitary polyhedron $(\mathbf{p} + \mathbf{n}, \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{p})$.

The volume of a tetrahedron $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$ is given by

$$V = \frac{1}{6} \det \begin{vmatrix} x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{vmatrix}. \tag{29}$$

Since the volume of any polyhedron may be considered to be the sum of the volumes of several tetrahedra, this can be straightforwardly calculated.

As we can see in Fig. 5 this procedure does not remove any undulations, but just moves the vertex to a new position, that is better centered in its star, improving the quality of the surface mesh, while preserving local volume. This improves the robustness of the undulation removal procedure presented below.

5.3. Undulation removal procedure

The vertex balance procedure is designed to produce a more homogeneous unstructured grid that represents the free surface or interface. However, it cannot remove the sub-grid undulations, and a smoothing procedure is required for the effective removal of these undulations.

Let **e** be an edge of the surface, and $\mathbf{v}_1$, $\mathbf{v}_2$ its vertices, as shown in Fig. 6. The smoothing procedure changes the position of $\mathbf{v}_1$ and $\mathbf{v}_2$ simultaneously, such that the local volume is preserved. Let $\mathbf{n}_1$ and $\mathbf{n}_2$ be

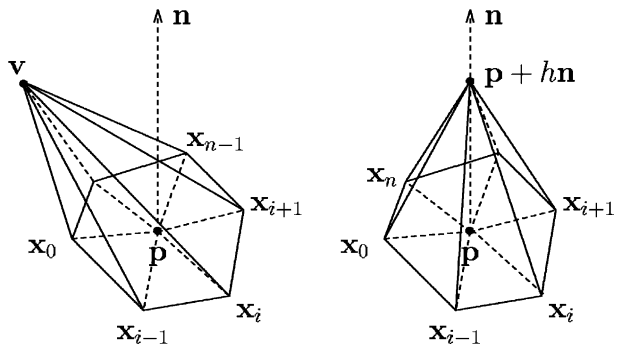

Fig. 5. A vertex with its star, before and after the vertex balance procedure has been applied.

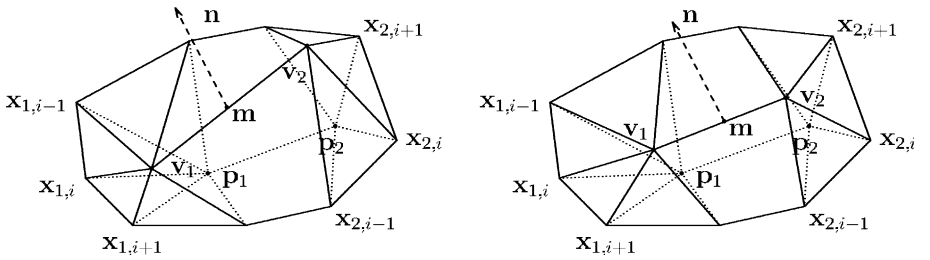

Fig. 6. Undulation removal procedure, showing an edge and its star.

the normal vectors computed by (26) at the vertices $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively. We consider a normal vector at the edge $\mathbf{e}$, given by the average of $\mathbf{n}_1$ and $\mathbf{n}_2$,

$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2}{\|\mathbf{n}_1 + \mathbf{n}_2\|}. \tag{30}$$

Let

$$\mathbf{m} = \tfrac{1}{2}(\mathbf{v}_1 + \mathbf{v}_2) \tag{31}$$

be the average of the vertices of the edge $\mathbf{e}$. We determine the heights h_1 and h_2 in the direction of this normal vector by

$$h_1 = \langle \mathbf{v}_1 - \mathbf{m}, \mathbf{n} \rangle \quad \text{and} \quad h_2 = \langle \mathbf{v}_2 - \mathbf{m}, \mathbf{n} \rangle. \tag{32}$$

Next, we compute the points $\mathbf{p}_1$ and $\mathbf{p}_2$ by

$$\mathbf{p}_1 = \mathbf{v}_1 - h_1\mathbf{n} \quad \text{and} \quad \mathbf{p}_2 = \mathbf{v}_2 - h_2\mathbf{n}. \tag{33}$$

Thus, we have two polyhedra $(\mathbf{v}_1, \mathbf{x}_{10}, \ldots, \mathbf{x}_{1n}, \mathbf{p}_1)$ with volume V_1 and $(\mathbf{v}_2, \mathbf{x}_{20}, \ldots, \mathbf{x}_{2m}, \mathbf{p}_2)$ with volume V_2, where $\mathbf{x}_{10}, \ldots, \mathbf{x}_{1n} \in S(\mathbf{v}_1)$ and $\mathbf{x}_{20}, \ldots, \mathbf{x}_{2m} \in S(\mathbf{v}_2)$ (see Fig. 6). We have to determine a new height h in the direction of $\mathbf{n}$, such that the local volume is preserved.

We can exploit linearity between the volumes of the polyhedra and the heights and write

$$V_1 + V_2 = V_1(h/h_1) + V_2(h/h_2) \tag{34}$$

or

$$h = \frac{V_1 + V_2}{V_1/h_1 + V_2/h_2}. \tag{35}$$

The new positions of the vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ are given by

$$\mathbf{v}_1 = \mathbf{p}_1 + h\mathbf{n} \quad \text{and} \quad \mathbf{v}_2 = \mathbf{p}_2 + h\mathbf{n}. \tag{36}$$

This undulation removal procedure is applied periodically to all the edges of the surface mesh with a period that has to be chosen such that the surface is sufficiently smooth, and the large scale undulations are not strongly affected. The frequency of applications is dependent on the problem, the grid resolution, and the time step.

6. Validation and numerical results

A number of tests was performed to validate the code and to assess its robustness and precision. First, we present some numerical results to validate the surface and interfacial tension computation, and to show the improvement of the TSUR-3D filter on the free surfaces or interfaces.

In the following subsections multi-fluid flow calculations are compared with their analytic solutions. In particular, we shall consider an oscillatory bubble, a bubble rising in a continuous phase and bubble coalescence. Finally, the splashing drop problem (with two phases) will be simulated to illustrate the capability and emphasize the robustness of the code.

6.1. Curvature evaluation

In order to validate the method used for the computation of the surface curvature at a local level, we report curvature computations at different points of the surfaces with constant and variable curvature. The numerical results were compared with the analytical values of the curvature using three different objects: a sphere, a cylinder and a torus. In fact, only the inner part of the torus is used, which possesses a critical line where the main curvatures are opposite.

As seen in Section 4, the calculation of the normal vector and the curvature depend on the size of the neighborhood chosen for each cell (S_δ and S_ε). Thus, the numerical tests were performed varying the size of these neighborhoods to see how much the error depends on this size. The results are shown in Table 1, where the radius size is a factor that is multiplied by $h = (\Delta x + \Delta y + \Delta z)/3$, where Δx, Δy and Δz are the mesh sizes in each direction. The error for the curvature approximation is estimated as

$$\mathrm{Error}_\kappa = \sqrt{\frac{\sum(\kappa_1 + \kappa_2 - \kappa_\mathrm{A})^2}{\sum(|\kappa_1| + |\kappa_2|)^2}}, \tag{37}$$

where κ_1 and κ_2 are the numerical curvature in each axis and κ_A is the analytical mean curvature. The summation is taken over the values computed on all the cells that contain marker particles of the objects representation. All the results were obtained on uniform regular grids and the surface representation of the objects can be seen in Fig. 7.

Table 1 shows the error and the standart deviation (SD) of curvature calculations on a sphere, cylinder and torus in $20 \times 20 \times 20$ grid with density of two marker particles per cell in each direction, corresponding to $h/R = 0.2$, where R is the characteristic radius of the object.

The radius given in Table 1 is directly related to the number of marker particles considered in the least squares approximation. Thus, the number of marker particles in the local approximation increases with the radius size, and the results tend to deteriorate as the radius increases substantially.

It can be seen that the method gives good results in simple geometries, such as the sphere and the cylinder, but the results tend to be worse in more complex geometries. However, the results are quite good

Table 1
Results of curvature calculations on a sphere, cylinder and torus, in $20 \times 20 \times 20$ grid with density of two marker particles per cell in each direction

Radius S_ε	Sphere		Cylinder		Torus	
	SD	Error	SD	Error	SD	Error
0.50	0.484789	3.860100	0.000000	59.678872	3.398686	0.400360
0.75	0.938420	0.707625	0.400466	0.499420	1.621055	0.155269
1.00	0.001898	0.004464	0.001000	0.003884	0.002280	0.011108
1.25	0.001823	0.007550	0.000922	0.005000	0.001199	0.010206
1.50	0.002819	0.011471	0.001453	0.009421	0.001606	0.012529
1.75	0.002632	0.015994	0.000505	0.011392	0.005074	0.018950
2.00	0.003212	0.021014	0.001571	0.016216	0.021629	0.030836
2.25	0.003482	0.026807	0.000524	0.018985	0.037758	0.039985
2.50	0.004296	0.033207	0.002314	0.026291	0.041469	0.043646

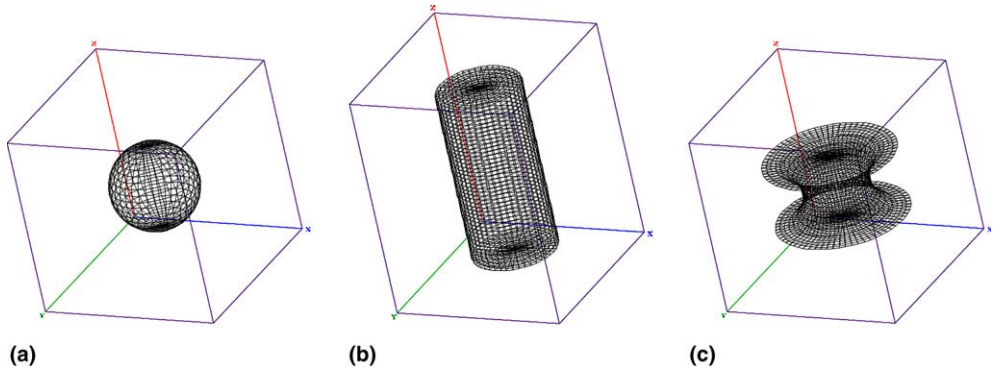

Fig. 7. Surface representation of the objects employed in the validation of the curvature calculation: (a) sphere; (b) cylinder; (c) torus.

for radii varying from 1.0 to 1.5, with errors about 1% in the torus and less than 1% in the other objects. These results also indicate the best choice for the radius S_ε.

6.2. Capillary pressure of spherical bubbles

To validate the computation of the interfacial tension and the capillary pressure, the problem of a spherical fluid (or bubble) of one phase is immersed into a fluid of another phase. This is then simulated using different grids. The densities of the fluids are $\rho_d = 1$ g/cm^3 for the bubble and $\rho_f = 0.5$ g/cm^3 for the continuous phase. The interface tension coefficient is $\sigma = 23.61$ dyn/cm, and the radius of the bubble is $R = 2$ cm. Fig. 8 shows the pressure jump at the interface, at the plane $y = 3$ cm. In the absence of viscous, gravitational, or external forces, surface or interfacial tension causes a static liquid bubble to become spherical. The Young–Laplace [3] equation for the capillary pressure is given by

$$\Delta p = \sigma\kappa = \sigma\left(\frac{1}{R_1} + \frac{1}{R_2}\right),$$

where R_1 and R_2 are the radii of the two perpendicular maximum circles of the sphere. In this case, the analytic value of the pressure jump at the interface is $\Delta p = 23.61$ dyn/cm^3, deducible directly from the Young–Laplace equation.

Numerical simulations were performed for five different uniform grids, with 20, 30, 40, 50 and 60 cells in each direction. Fig. 9 displays the numerical capillary pressure converging to the analytic value given by the Young–Laplace equation (a), and the error as the grid is refined (b).

These numerical results are accurate with an error in the Euclidean norm smaller than 1.2% for the coarsest grid, reduced to about 0.2% when the grid is refined. The convergence of the method for interfacial tension calculation is quadratic, or order 2, as can be seen in Fig. 9(b).

6.3. Parasitic currents

As stated by Tryggvason et al. [29], in addition to the convergence difficulties sometimes encountered at high density ratios, the pressure solution can cause other difficulties. If the surface tension coefficient is high, and if the representation of the surface forces on the grid has any significant anisotropy, unphysical velocities can be generated. These velocities, sometimes called "parasitic currents", are usually small.

In the absence of gravity, a spherical static bubble immersed in a continuous phase should remain exactly stationary, and the velocities should be exactly zero. However, small fluctuations in the surrounding

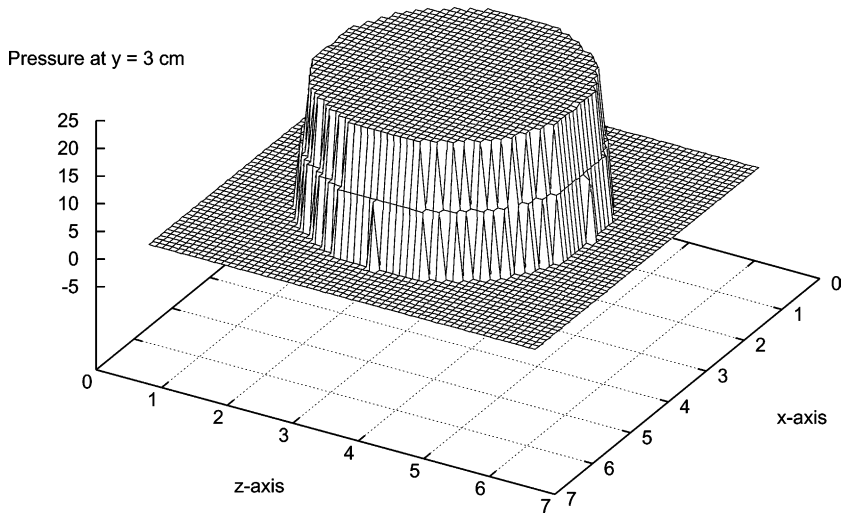

Fig. 8. Capillary pressure of a spherical bubble in a $60 \times 60 \times 60$ grid, at $y = 3$ cm.

fluid near the bubble can introduce recirculation in the flow. These unphysical velocities depend strongly on many factors such as grid resolution, viscosity, surface tension, etc.

We simulated this problem using a bubble with radius $R = 0.25$ in an unitary domain, with $25 \times 25 \times 25$ and $50 \times 50 \times 50$ computational cells in the whole domain. We used the same capillary-viscous length as Tryggvason, given by $R_v = 0.002$, where $R_v = \rho v^2/\sigma$, obtaining a maximum velocity of $O(10^{-3})$ in the $25 \times 25 \times 25$ grid against $O(10^{-4})$ obtained by the two-dimensional version of the Tryggvason's code. For the fine grid ($50 \times 50 \times 50$) we obtained a maximum velocity of $O(10^{-4})$ against $O(10^{-5})$ from the 2D Tryggvason's code.

These results show that the parasitic currents generated by our code are quite small, and do not degrade the solutions when the surface tension coefficient is high.

6.4. Sessile drop

A simulation of a single phase sessile drop was performed to validate the surface tension and curvature calculations. The steady state numerical solutions were obtained from the asymptotic steady state solutions of the transient solution, starting from a spherical drop initially at rest on a plane. Highly accurate solutions were obtained by the numerical integration of the equations for the equilibrium position of an axisymmetric free surface liquid drop. These are, in non-dimensional form, given by

$$\frac{\mathrm{d}\theta}{\mathrm{d}s} = \mathrm{Bo}(p - z) - \frac{\cos\theta}{r}, \quad \frac{\mathrm{d}r}{\mathrm{d}s} = -\sin\theta \quad \text{and} \quad \frac{\mathrm{d}z}{\mathrm{d}s} = \cos\theta, \tag{38}$$

where θ is the angle between the surface outward normal and the r axis, s is the coordinate along the surface, $\mathrm{Bo} = \rho g L^2/\sigma$ is the Bond number, and $p = \hat{p}/\rho g L$ is a non-dimensional pressure, where here $\hat{p}$ is the dimensional reference pressure.

A fourth-order Runge–Kutta method has been used to integrate (38), using an integration step $\Delta s = 10^{-4}$. Hence the solutions can be regarded as being very accurate, aside from a region in the vicinity of the axis $r = 0$, where the singularity $\sin\theta/r$ may degrade the accuracy. To avoid integrating close to the singularity, we start the integration from the point of maximum r and integrate up to the meniscus, and

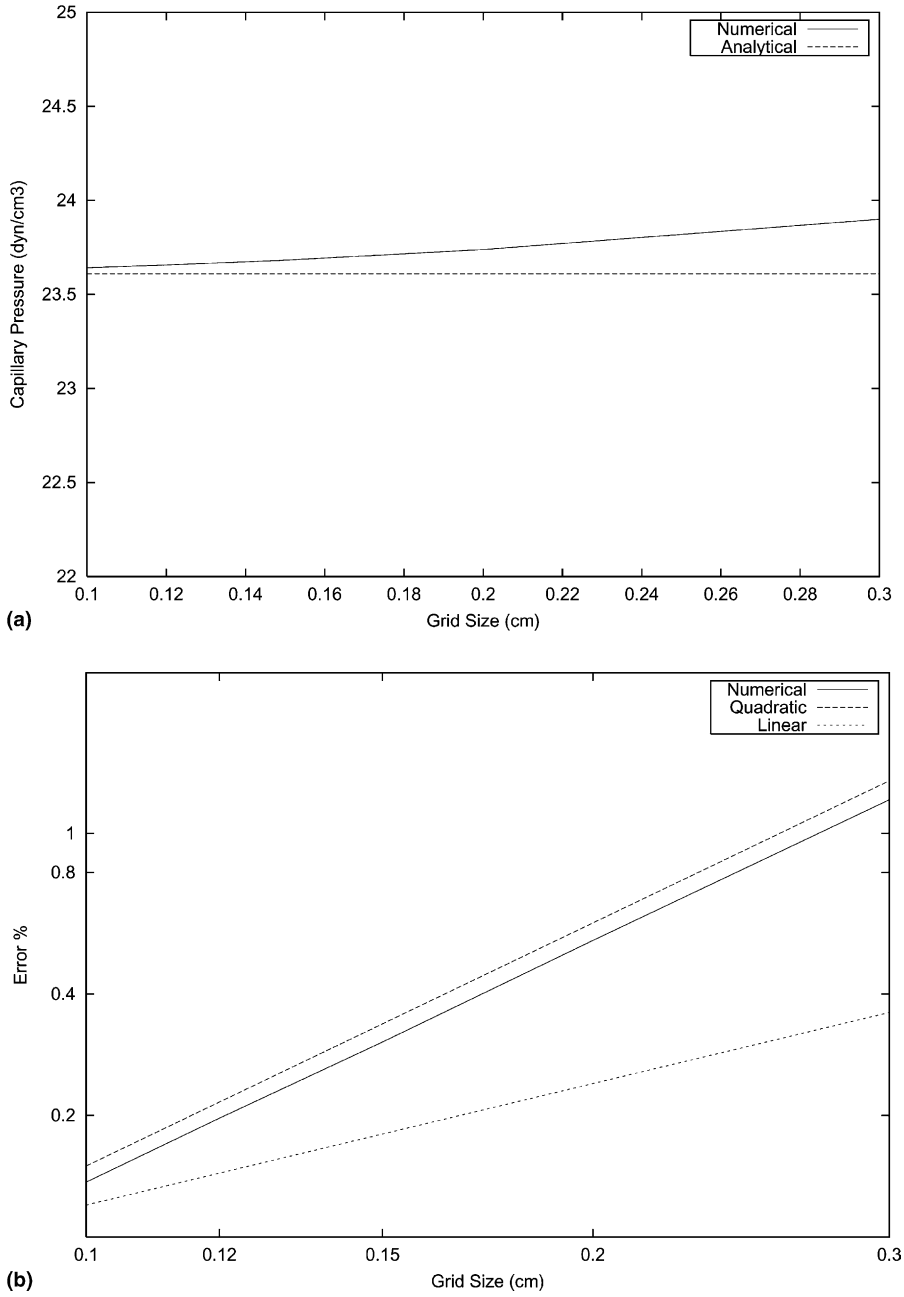

Fig. 9. Numerical capillary pressure displaying convergence to the analytical capillary pressure (a) and a log scale graph showing the second-order convergence (b).

then down to the contact point. The loss of accuracy of this "exact" solution is therefore restricted to a very small region in the vicinity of the axis, and is therefore, for our purposes, of no concern. A quantitative comparison of the results can be seen in Fig. 10. Fig. 10(a) shows good agreement between the "exact" and

numerical computations of this paper. The above results were obtained using $28 \times 28 \times 28$ computational cells.

The largest departure from the "exact" solution is observed at the top of the drop in Fig. 10(a), where the numerical solution shows some short wavelength (sub-grid) undulations. The occurrence of undulations has been observed to become more pronounced as the Reynolds number is increased, thus requiring the

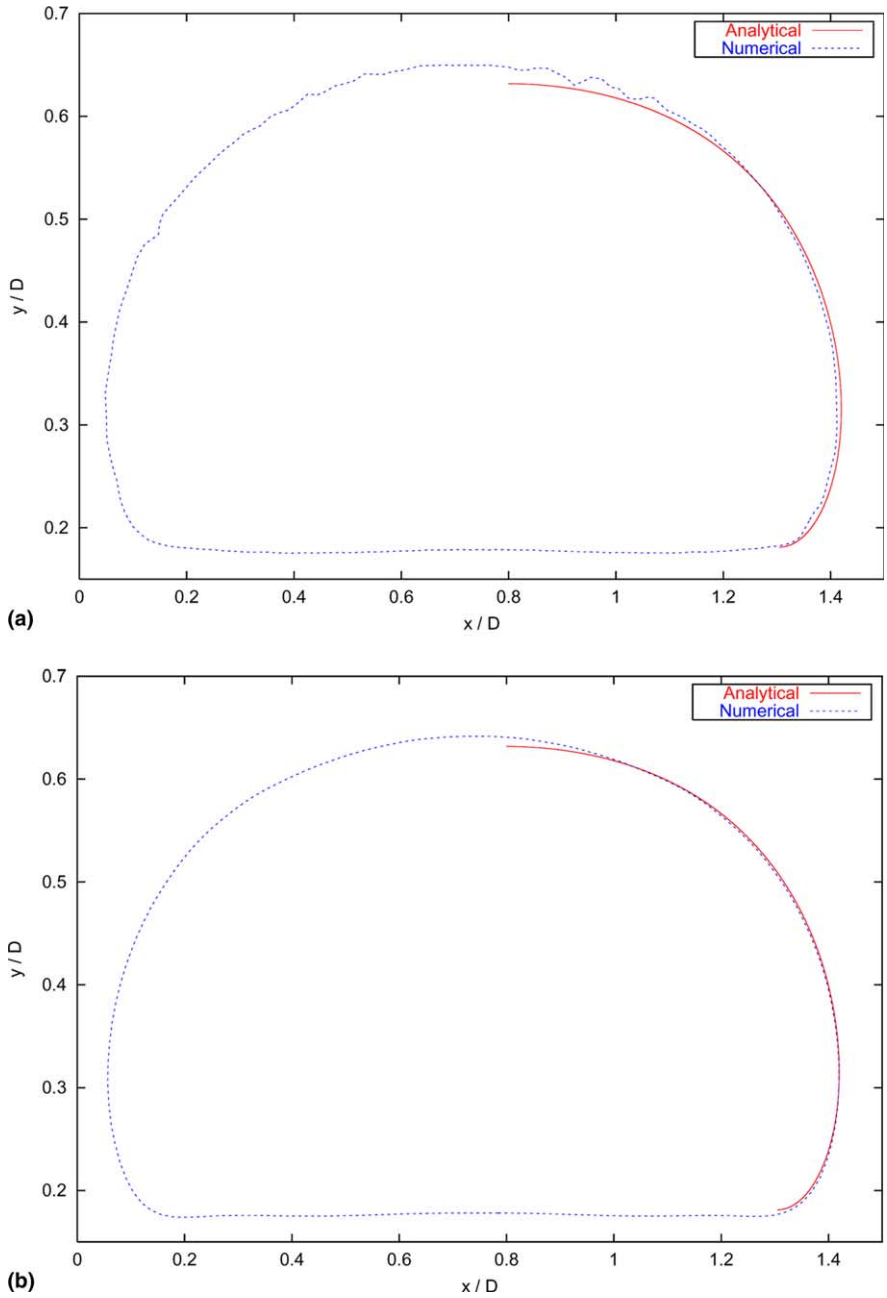

Fig. 10. Comparison between "exact" and numerical solution: without applying the TSUR-3D filter (a), and with the TSUR-3D filter (b).

application of a sub-grid filter when simulating flows with higher Reynolds number (above 50). The application of a sub-grid filter improves the results of the method, as can be seen in Fig. 10(b).

6.5. Oscillation of a bubble

To verify that the transfer between the surface energy and kinetic energy is correctly accounted for, which is important when the flow is dominated by surface tension effects, we validated the method on an ellipsoidal oscillating bubble, for which an analytical solution for the oscillation frequency exists.

This problem consists of simulating an ellipsoidal bubble immersed in a continuous phase without a gravity field. The bubble has a small initial perturbation with respect to its equilibrium spherical form and, driven by the interfacial forces, it tends to oscillate. The non-dimensional parameters chosen for the simulation were $\sigma = 10$, $\rho_d = 100$ and $\mu_d = 0.35$ for the bubble and $\rho_f = 1$ and $\mu_f = 0.001$ for the surrounding fluid. The bubble's initial radius (R) was 1 and the initial amplitude of the perturbation corresponds to 2.5% of its radius. The calculations were made in a domain of size $4R \times 4R \times 4R$, discretized by both $20 \times 20 \times 20$ and $40 \times 40 \times 40$ computational grids. The results obtained were compared with the analytic solutions [7,8,30], displayed in Fig. 11. The analytic expression for the frequency oscillation of a bubble in an infinite domain is given by

$$\omega^2 = \frac{24\sigma}{(3\rho_d + 2\rho_f)R^3}.$$

Assuming the viscous effects to be small, the amplitude will decay as

$$a(t) = a_0 e^{-t/\tau},$$

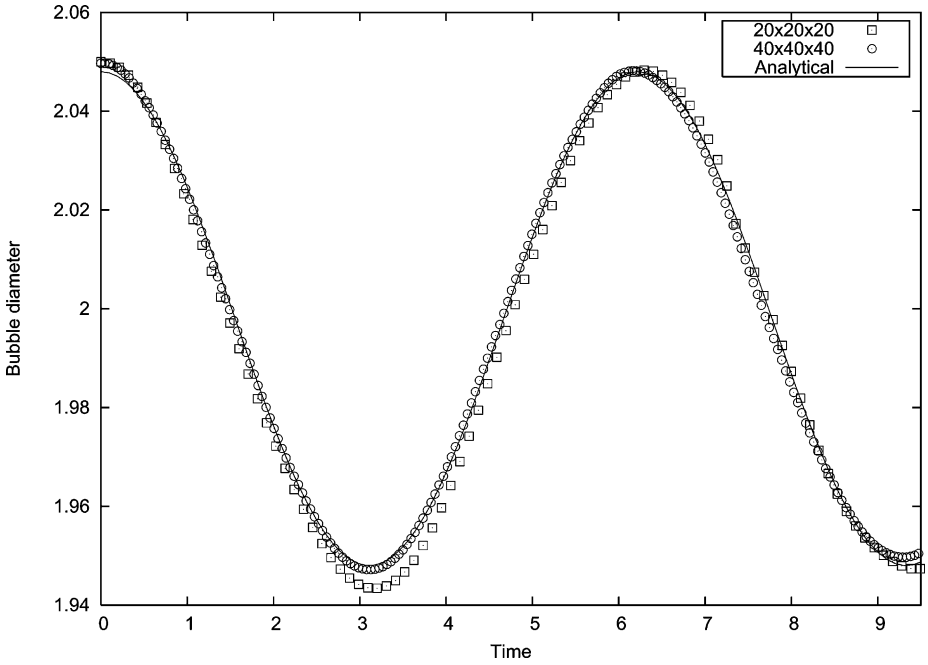

Fig. 11. Oscillation of the bubble diameter as a function of non-dimensional time. It was found that the period was in error by 2.2% in the coarse grid and 1.2% in the fine grid.

if the effect of the surrounding fluid is neglected, where $\tau = R/5v$ and a_0 and v are the initial amplitude and kinematic viscosity, respectively.

Fig. 11 shows a comparison between the numerical result and the analytical curve, given by

$$y(t) = y_0 + a_0 e^{-t/\tau}\cos(\omega t),$$

where y_0 and a_0 are the initial oscillating axis position and the amplitude.

It can be seen that the numerical results are in good agreement with the analytic expression for the bubble diameter. Using the extreme points to estimate the period of oscillation, we found the discrepancy between the numerical and the analytical value for the period is about 2.2% in the coarse grid and about 1.2% in the fine grid.

6.6. Bubble rising in a continuous phase

Rising bubbles are classical examples that may be used to validate multi-fluid flow simulations. Bubbles with lower density than the surrounding fluid tend to rise, due to the buoyancy effects resulting from the pressure gradient caused by gravity.

In this paper, two numerical results of rising bubbles with low and moderate Reynolds numbers are presented. The results, obtained using stationary domains with no-slip boundary conditions, are compared with results published by Esmaeeli and Tryggvason [7,8] for bubbles rising in periodic domain.

6.6.1. Low Reynolds numbers

The nondimensional parameters, extracted from Esmaeeli and Tryggvason [7], are given by $Eo = 1.0$, $N = 10^{3/2}$, $\gamma = 0.05$ and $\lambda = 0.05$, where $Eo = \rho_0 g D^2/\sigma_0$ is the Eötvös number, $N = \rho_0^2 D^3 g/\mu_0^2$ is the Galileo or Archimedes number, γ is the density ratio between the bubble and the fluid (ρ_b/ρ_f) and λ is the viscosity ratio between the bubble and the fluid (μ_b/μ_f). These parameters can be obtained by using a 1.9 mm diameter bubble immersed in engine oil, with $\sigma_I = 0.03$ N/m, $\rho_f = 880$ kg/m^3 and $\mu_f = 0.21$ N s/m^2. The lengths were nondimensionalized by the diameter $D = 1.9$ mm and the velocities by $U = \sqrt{gD}$.

To compare the results with [7], it is necessary to compute the rising velocity and the centroid position of the bubble at each time step. As the bubble surface is explicitly marked by moving particles, it is more convenient to change the volume integrals into surface integrals using the divergence theorem. Thus, if the bubble volume is given by

$$V_b = \int_{V_b} \mathrm{d}V = \frac{1}{3}\oint_{S_b} \mathbf{x}\cdot\mathbf{n}\,\mathrm{d}S,$$

then the centroid position is

$$\mathbf{x}_c = \frac{1}{V_b}\int_{V_b} \mathbf{x}\,\mathrm{d}V = \frac{1}{2V_b}\oint_{S_b} (\mathbf{x}\cdot\mathbf{x})\mathbf{n}\,\mathrm{d}S,$$

and, using the continuity equation, the rising velocity is given by

$$W_b = \frac{1}{V_b}\int_{V_b} w\,\mathrm{d}V = \frac{1}{V_b}\oint_{S_b} (z\mathbf{u})\cdot\mathbf{n}\,\mathrm{d}S.$$

Fig. 12 shows the results obtained on grids with $16\times16\times32$ and $32\times32\times64$ cells. In this figure small oscillations in the rising velocity can be observed. These were eliminated when the grid was refined. The rising velocity oscillates due to the pressure gradient in the coarse grid when the bubble moves from cell to

cell. At the terminal velocity of the bubble, the Reynolds number obtained (see Fig. 12(a)) is quite close to 1.4. This is the same value obtained by Esmaeeli and Tryggvason [7].

Fig. 13 shows the volume variation of the bubble in nondimensional time. For this case, the error found in mass conservation is about 0.5% in the fine grid. This loss of mass is due to the unstructured mesh

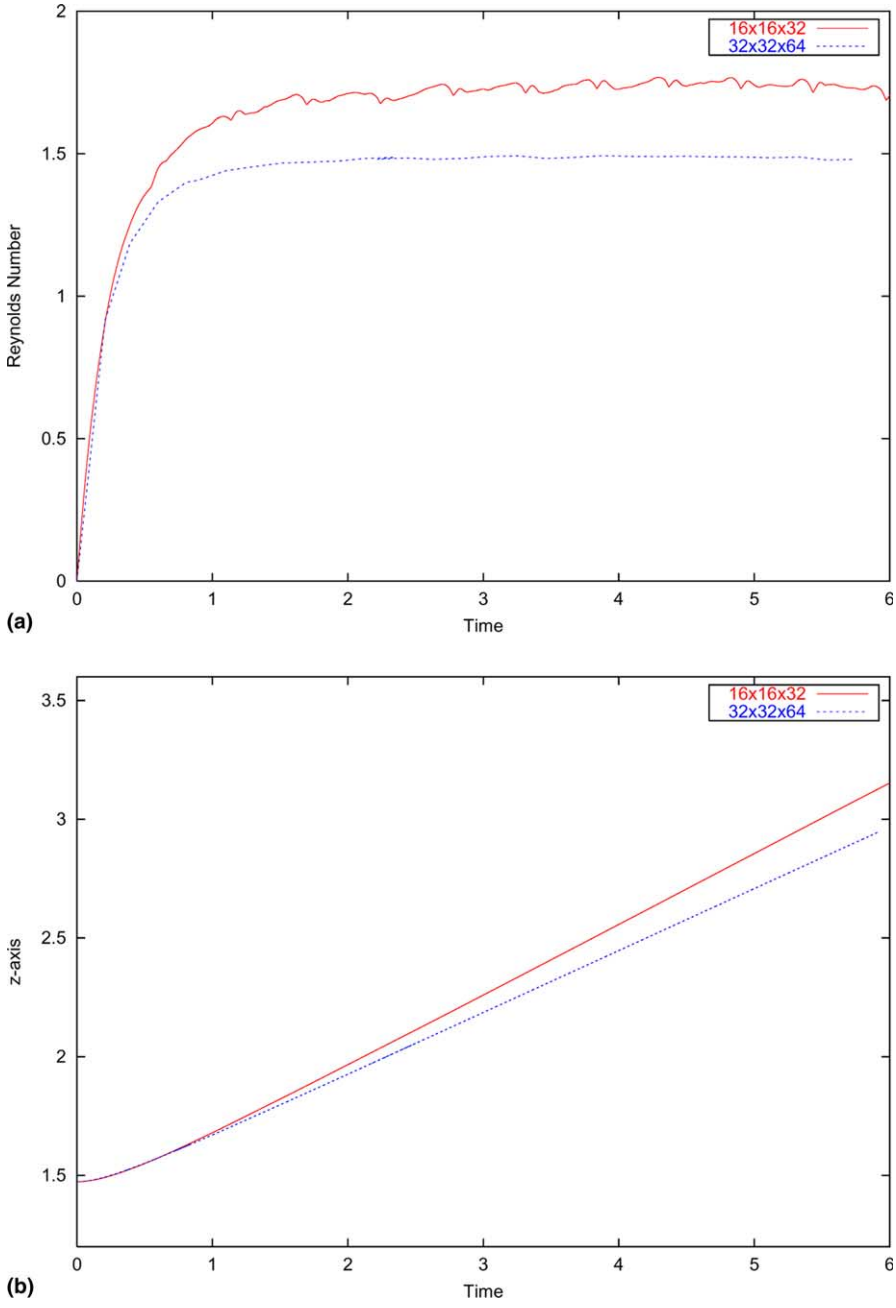

Fig. 12. Rising Reynolds number (a) and the bubble centroid position (b) plotted against nondimensional time on $16 \times 16 \times 32$ and $32 \times 32 \times 64$ grid cells (low Reynolds number case).

control procedure, which inserts and deletes marker particles from the surface of the bubble every time step. In fact, the deletion procedure usually reduces the volume of the bubble, and this effect is more pronounced in finer grids where the time step is smaller, as displayed in Fig. 13. All simulations were performed on a Pentium III machine with 1.0 GHz. The CPU times for rising bubble simulations (low Reynolds case, when the method is at its most inefficient) are as follows: 17 h 20 min for grid size $16 \times 16 \times 32$ and 540 h for grid size $32 \times 32 \times 64$.

6.6.2. Moderate Reynolds numbers

For this simulation, the nondimensional parameters extracted from Esmaeeli and Tryggvason [8] are $Eo = 2$, $N = 894.4$, $\gamma = 0.1$ and $\lambda = 0.1$. These can be obtained from $D = 2.6$ mm diameter bubble, with $\mu_f = 0.0125$ N s/m^2, $\rho_f = 880$ kg/m^3 for the continuous phase (fluid) and interfacial tension coefficient given by $\sigma_I = 0.03$ N/m. The lengths were nondimensionalized by the diameter D and the velocities by $U = \sqrt{gD}$.

As in the low Reynolds number case, the numerical results were compared with those obtained by Esmaeeli and Tryggvason [8], by calculating the rising velocity and centroid position of the bubble. The graphs of these quantities can be seen in Fig. 14. As in the previous simulation, the results show small oscillations in the rising velocity that vanish when the grid is refined.

The results obtained by Esmaeeli and Tryggvason [8] show that the fully developed Reynolds number is about 20.5, while the result obtained here is $Re \approx 25$. This discrepancy is due to some differences in the boundary conditions between the two models. However, the results obtained are qualitatively similar.

6.6.3. Rising bubble with high density ratio

This example was simulated to show that the code can deal with flows with high density ratios using a sharp interface. The parameters for this simulation are $Eo = 2$, $N = 894.4$, $\gamma = 0.001$ and $\lambda = 0.5$. As in the

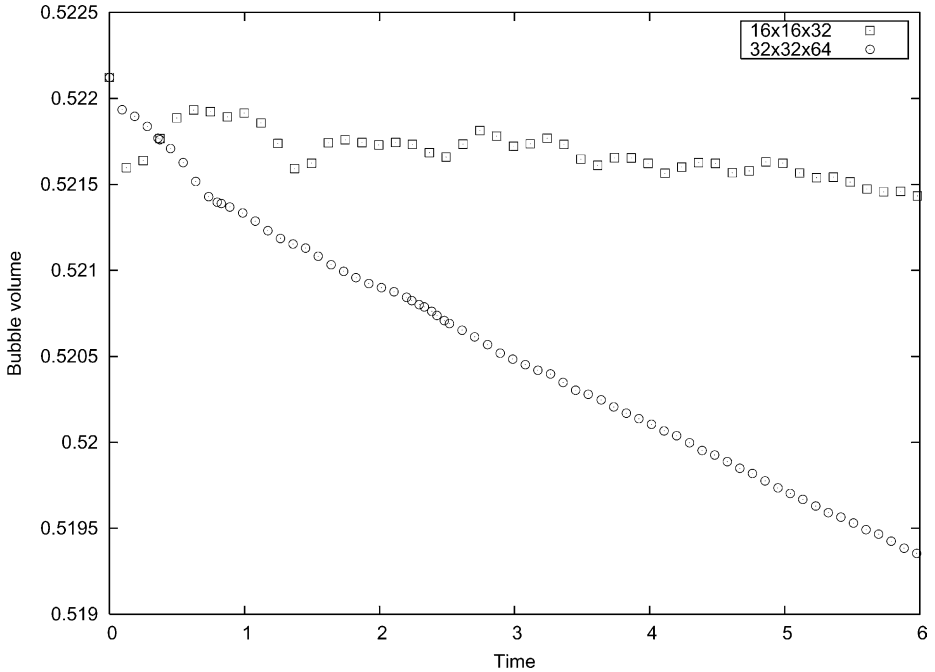

Fig. 13. Bubble volume against nondimensional time on $16 \times 16 \times 32$ and $32 \times 32 \times 64$ grid cells. The error in mass conservation is about 0.5% in the fine grid.

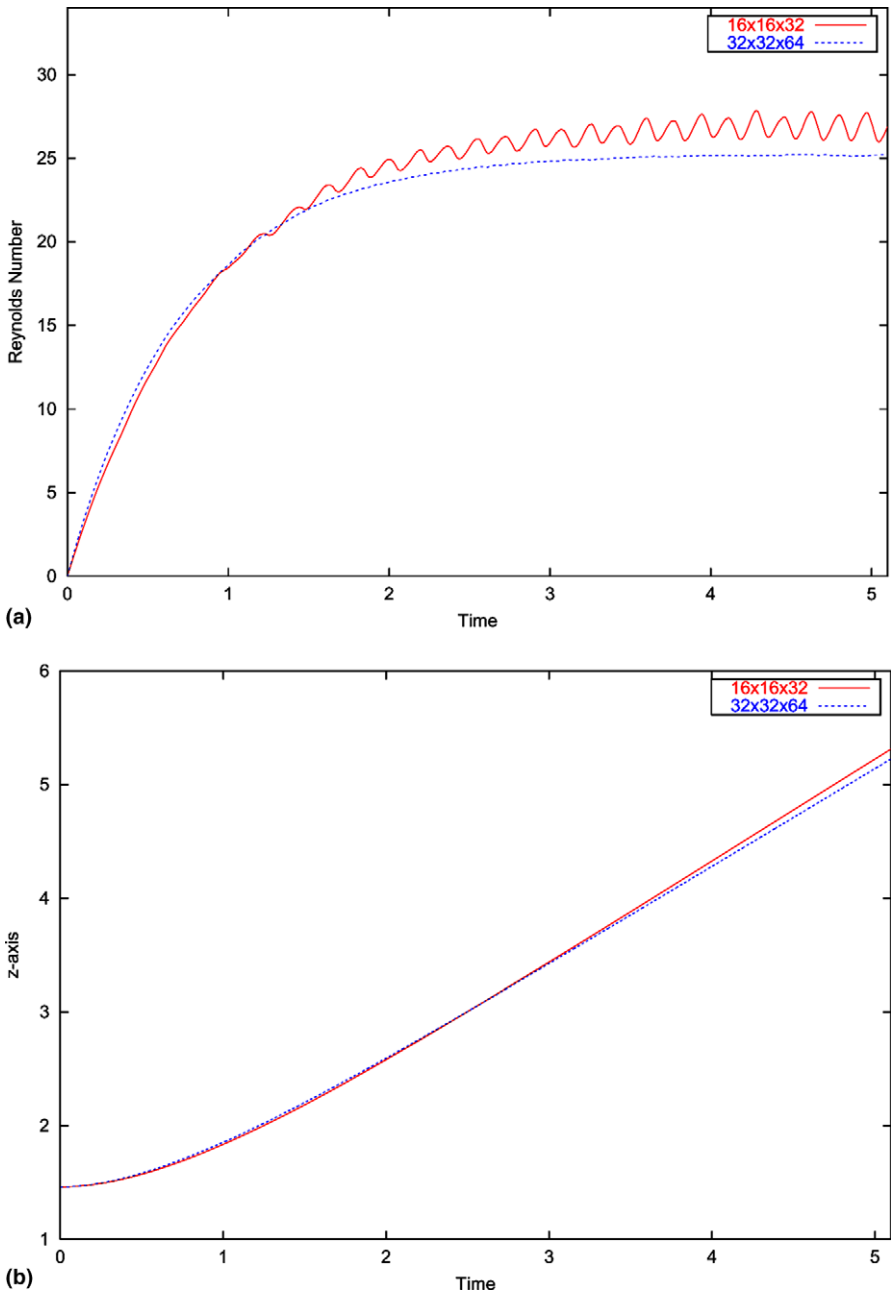

Fig. 14. Rising Reynolds number (a) and the bubble centroid position (b) plotted against nondimensional time on $16 \times 16 \times 32$ and $32 \times 32 \times 64$ grid cells (moderate Reynolds number case).

previous simulations, we considered $D = 2.6$ mm diameter bubble, with $\mu_f = 0.0125$ N s/m^2, $\rho_f = 880$ kg/m^3 for the continuous phase (fluid) and interfacial tension coefficient given by $\sigma_I = 0.03$ N/m. The lengths were nondimensionalized by the diameter D and the velocities by $U = \sqrt{gD}$.

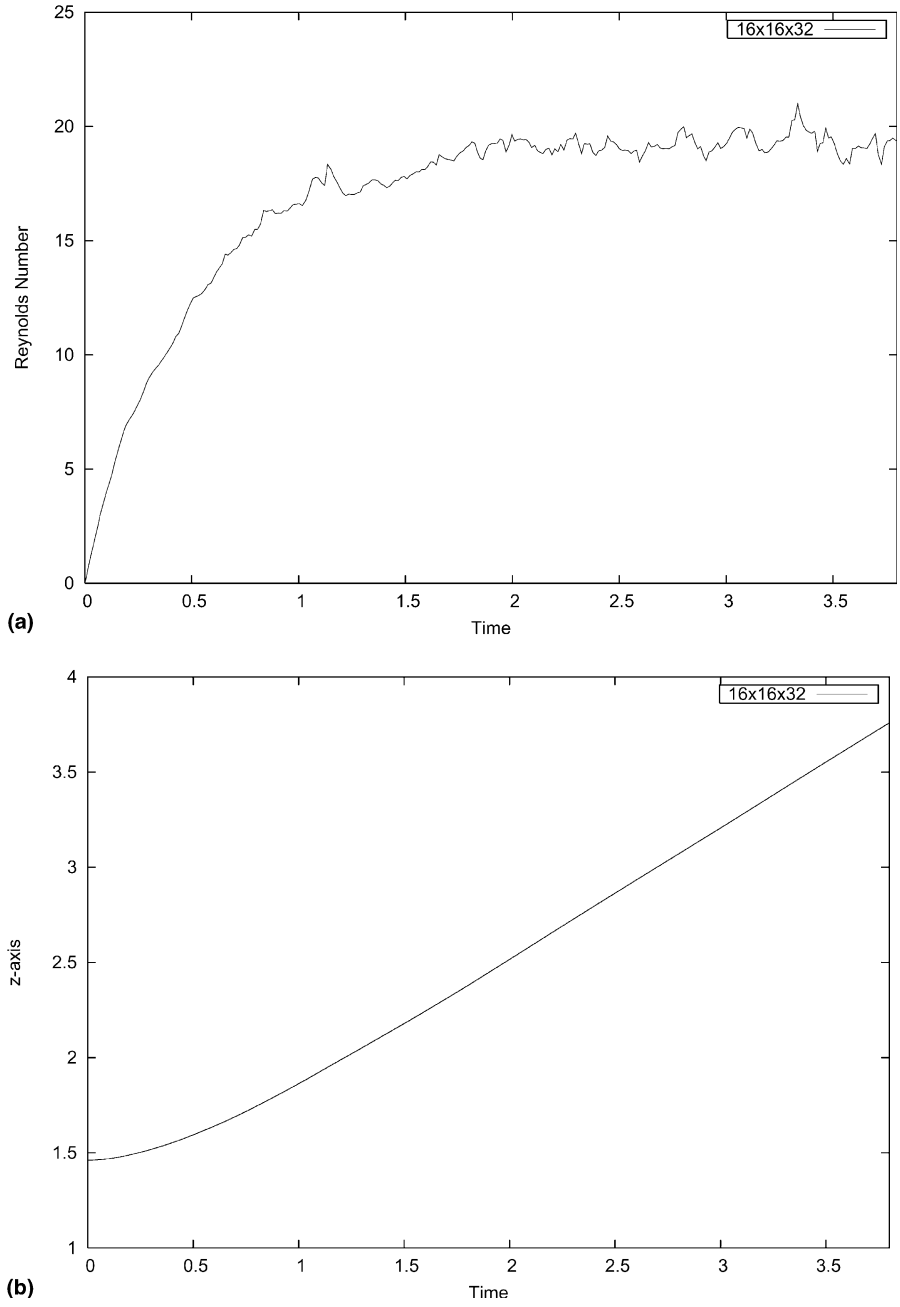

Fig. 15. Rising Reynolds number (a) and the bubble centroid position (b) plotted against nondimensional time on $16 \times 16 \times 32$ grid cells for the high density ratio case.

In Fig. 15 the same kind of oscillations can be seen with increasing Reynolds number, due to the coarse grid used in this simulation ($16 \times 16 \times 32$ computational cells). This figure displays similar behavior to the previous simulations, as expected.

6.7. Rising of two different bubbles

This example shows the simulation of the three-fluid flow of two bubbles with different densities, viscosities and diameters, rising in a third phase with a greater density.

As in the previous example, the bubbles rise due to buoyancy forces. The nondimensional parameters are: $Re = 55.75$, $Fr = 1$, $We = 4.6$. For the continuous phase, $\rho_f = 880$ kg/m^3 and $\mu_f = 0.0125$ N s/m^2. For the larger bubble, $D_1 = 4$ mm, $\rho_1 = 88$ kg/m^3 and $\mu_1 = 0.00125$ N s/m^2, with $Eo_1 = 4.6$. Finally, for the smaller bubble, $D_2 = 2$ mm, $\rho_2 = 176$ kg/m^3 and $\mu_2 = 0.0025$ N s/m^2, with $Eo_2 = 1.15$. The surface tension coefficient for all interfaces is $\sigma_I = 0.03$ N/m. The lengths and velocities were nondimensionalized using $D_1 = 4$ mm and $U = \sqrt{gD_1}$, respectively. The grid used in this example was $32 \times 32 \times 64$ cells.

Figs. 16 and 18 show the time evolution of the flow. It can be seen that the larger bubble undergoes a larger deformation than the smaller bubble, due to its larger Eötvös number. Fig. 17 shows a small deflection in the trajectory of the smaller bubble due to the interaction with the larger bubble as it passes by; the deflection is more readily seen in Fig. 18.

6.8. Bubble coalescence

Another classical example of multi-fluid flows is the coalescence of two bubbles in a continuous phase. The two bubbles are positioned in-line at a distance of 0.4 mm apart. The diameter of both bubbles is $D = 2.6$ mm, the velocities of each bubble are nondimensionalized using $U = \sqrt{gD} = 0.15$ m/s, the interfacial tension coefficient is $\sigma_I = 5.8 \times 10^{-4}$ N/m, and the nondimensional constants $Re = 30$ and $Eo = 100$ are the Reynolds and Eötvös numbers, respectively. For the continuous phase, $\rho_f = 880$ kg/m^3, $\mu_f = 0.0125$ N s/m^2, and for the bubbles, $\rho_d = 440$ kg/m^3, $\mu_d = 0.00625$ N s/m^2. Fig. 19 shows the transient solution of the interface, with the coalescence of both bubbles, starting from the bubbles aligned vertically. This simulation takes about 76 h and 30 min in a Pentium III machine with 1.0 GHz.

The same flow parameters were used to simulate the rise of two bubbles that are initially not aligned. Fig. 20 shows the three-dimensional rendering of the transient solution, in which the bottom bubble chases the top bubble until coalescence occurs. This is due to the lower pressure close to the bottom of the top bubble. These results are in good agreement when compared with the numerical results obtained by Chen and Li [6] and the experimental results obtained by Narayanan et al. [16]. To be more specific, the above experimental and the numerical results show that, due to the effect of the velocity field around the leading bubble, the following bubble is stretched and a spherical-cup-shaped leading bubble is observed. In other words, the leading bubble induces a deformation in the following bubble, giving it a pear-like behavior. Fig. 19 clearly shows this same behavior.

In the case of nonaxisymmetric three-dimensional merging, the results can be compared to results published by Shin and Juric [21], where they report that "the bubbles deform considerably as they rise. The

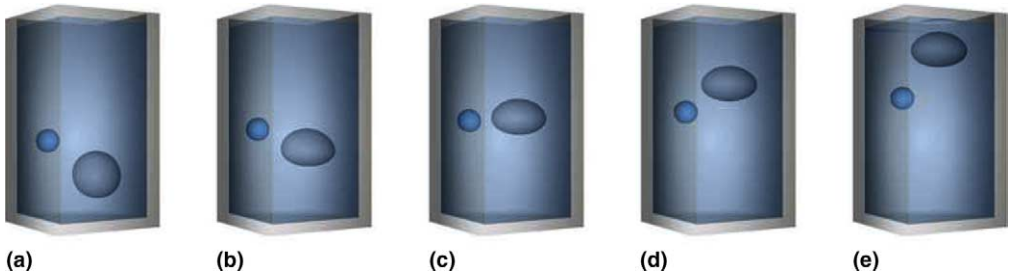

Fig. 16. Two bubbles rising in a continuous phase, at different times. The nondimensional parameters are $Re = 55.75$, $Fr = 1$ and $We = 4.6$: (a) 0.01 s; (b) 0.03 s; (c) 0.05 s; (d) 0.07 s; (e) 0.09 s.

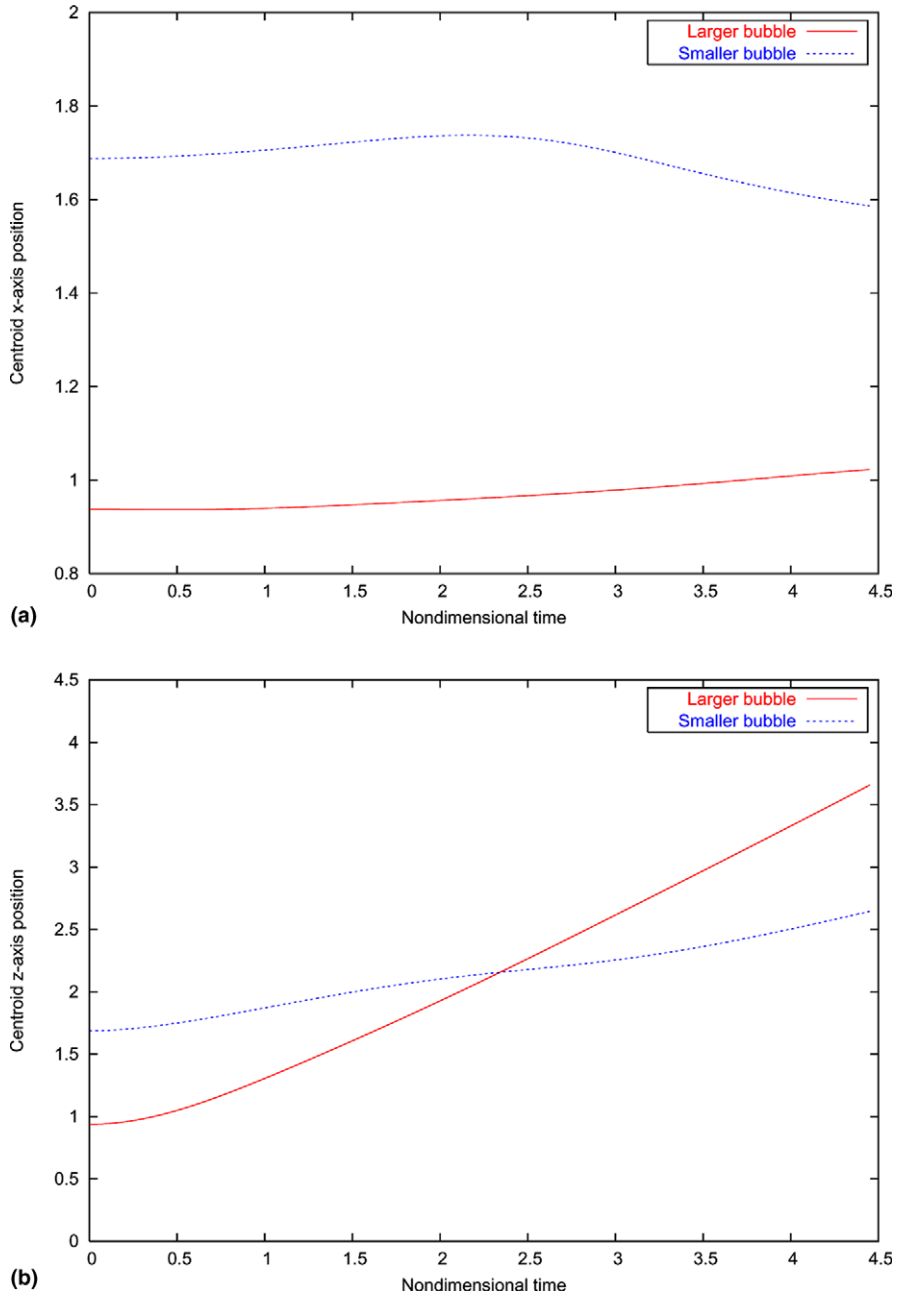

Fig. 17. Rising of two different bubbles: (a) coordinate x and (b) coordinate z of the bubbles centroid position plotted against non-dimensional time.

bottom of the top bubble folds upward and deforms the lower bubble into a pear shape, pointing towards the top bubble. The top bubble continues to deform into a hemispherical shell. The lower bubble deforms into a more cylindrical shell and ultimately gets sucked upward merging with the top bubble while for a short time trailing a thin tail". Fig. 20 clearly shows this same type of behavior.

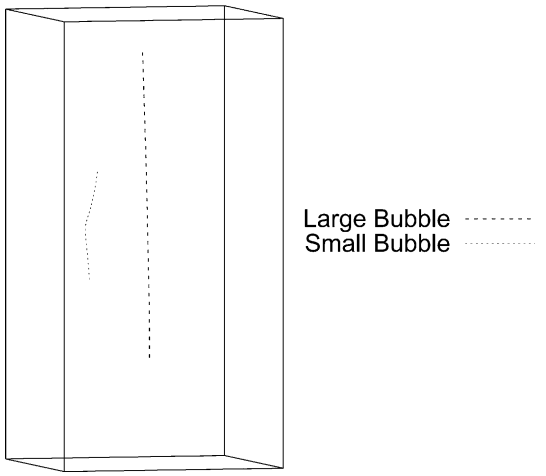

Fig. 18. Rising of two different bubbles: bubbles centroid three-dimensional trajectory.

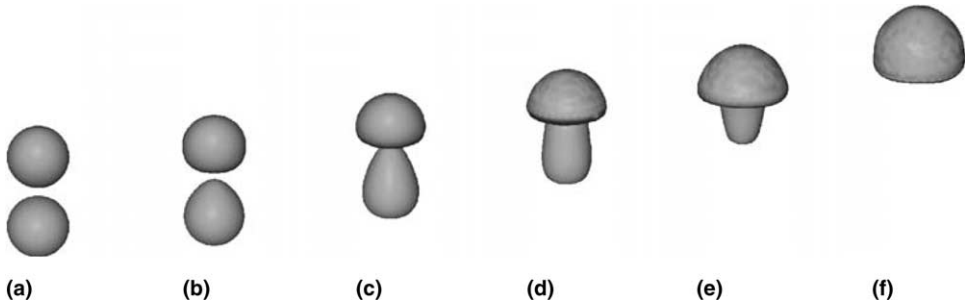

Fig. 19. Transient solution of bubble coalescence in a continuous phase, with in-line bubbles. The nondimensional parameters are $Re = 30$, $Fr = 1$ and $Eo = 100$: (a) 0.0 s; (b) 0.03 s; (c) 0.06 s; (d) 0.09 s; (e) 0.12 s; (f) 0.15 s.

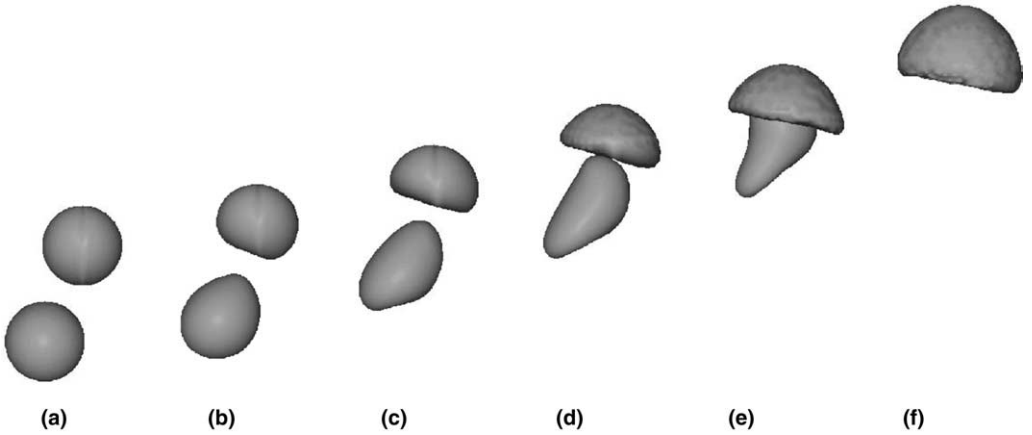

Fig. 20. Transient solution of bubble coalescence in a continuous phase, when the bubbles are not aligned. The nondimensional parameters are $Re = 30$, $Fr = 1$ and $Eo = 100$: (a) 0.0 s; (b) 0.03 s; (c) 0.06 s; (d) 0.09 s; (e) 0.12 s; (f) 0.15 s.

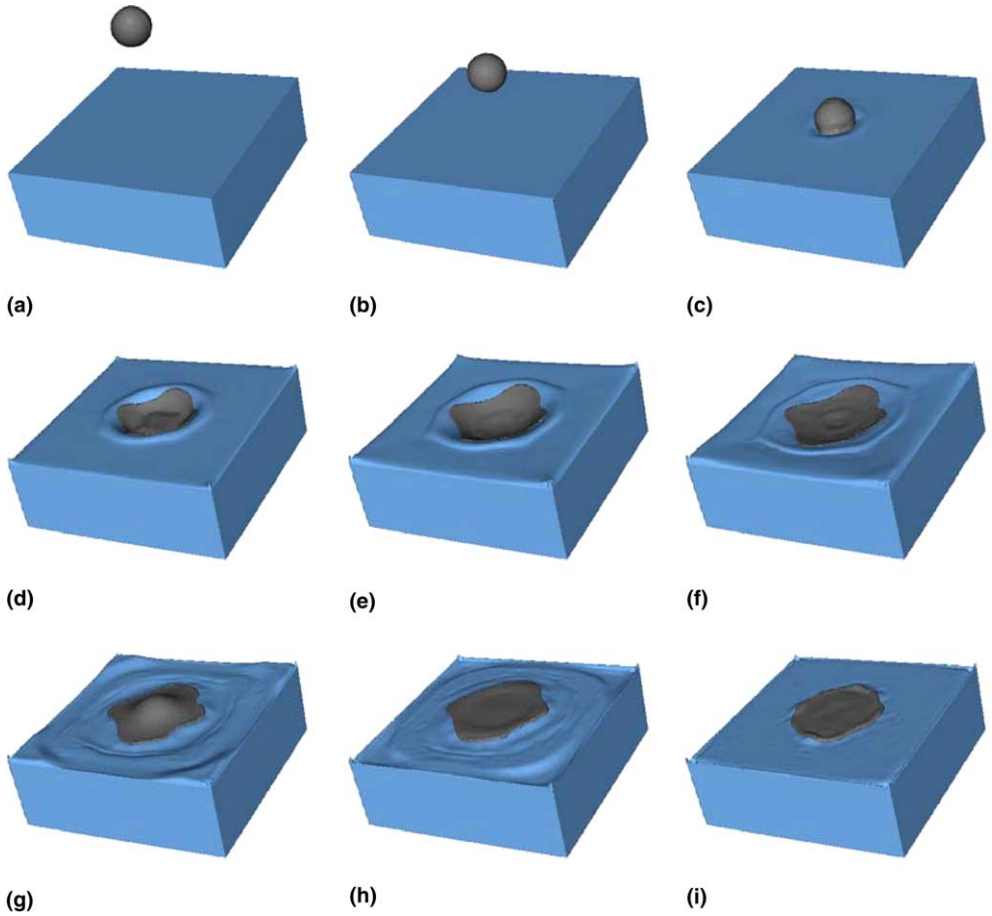

Fig. 21. Transient solution of the splashing drop falling into a free surface of a heavier and more viscous fluid. The nondimensional parameters are $Re = 30$, $Fr = 1$ and $We = 50$: (a) $t = 0$ s; (b) $t = 0.03$ s; (c) $t = 0.04$ s; (d) $t = 0.05$ s; (e) $t = 0.06$ s; (f) $t = 0.07$ s; (g) $t = 0.09$ s; (h) $t = 0.12$ s; (i) $t = 0.25$ s.

6.9. Two-fluid splashing drop

Splashing drops are good examples to illustrate the robustness of the free surface flow codes. In this particular case, a two-fluid splashing drop was simulated, whereby a drop falls into a container of a quiescent fluid of a different phase resulting in a splash.

The drop phase was taken to be less viscous and lighter than the continuous phase. The nondimensional parameters were given by: $Re = 30$, $Fr = 1$ and $We = 50$. For the drop phase, we set $\rho_g = 440$ kg/m^3 and $\mu_g = 0.00625$ N s/m^2; for the continuous phase, we used $\rho_f = 880$ kg/m^3 and $\mu_f = 0.0125$ N s/m^2. The interfacial tension coefficient was $\sigma_I = 0.001167$ N/m, and the free surfaces are driven by surface tension forces, with the same coefficient. Again, we used the drop diameter D and the reference velocity $U = \sqrt{gD}$ to nondimensionalize the lengths and velocities.

Fig. 21 shows the transient solution of the flow in a $40 \times 40 \times 40$ grid. In this figure we are able to observe the surface tension effects acting over the free surfaces. Fig. 22 shows the rendering of the solution using pseudo-color to indicate velocity components and pressure over the free surfaces. Fig. 22 displays the

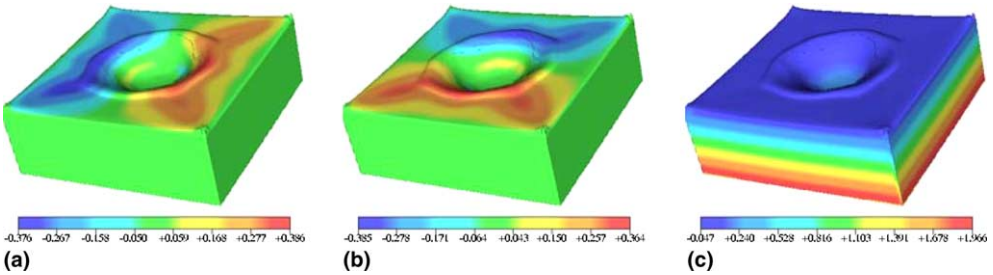

Fig. 22. Three-dimensional color scale rendering of the solution at time $t = 0.06$ *s*: (a) *x*-direction velocity; (b) *y*-direction velocity; (c) pressure.

x-direction velocity, the *y*-direction velocity and the pressure at time $t = 0.06$ s. We can see that the velocity field remains symmetric after the drop impact.

This simulation is an important indicator of the robustness of the code, where we can see the interaction between the free surfaces and the interface. The code has the capability to decide when fronts should be treated as free surfaces or as interfaces, using the information stored in its internal cellular representation.

7. Conclusions

In this work, a method for simulating incompressible multi-fluid flows with surface tension was described. This method was based on the GENSMAC-3D front-tracking method [28], using finite-difference schemes to discretize the governing equations, as in [18]. Surface and interfacial tensions were considered, and the required curvature on the interface was computed by a new technique of geometrical surface approximation [4,13,22,24].

A number of numerical simulations was used to validate the method presented in this paper. The simulation of static bubbles showed good agreement with the analytic solution, with errors less than 1.2% indicating that the curvature calculation procedure is accurate. A numerical convergence test was performed indicating quadratic convergence of the curvature calculation.

A sessile drop simulation was also carried out to verify the accuracy of the surface tension approximation method. The results were compared with the analytic solution for the steady state axisymmetric case showing good agreement. Two types of simulations were performed. The first one shows small undulations over the free surface that are unphysical and due to the discretization of the domain. Since the surface tension method acts on the scale of a cell it cannot eliminate these undulations in scales smaller than a cell. Thus a geometric filter was developed. The filter, called TSUR-3D [23,24], smooths the free surface and interfaces, while conserving the mass of the fluid. The second type of simulation was made with the TSUR-3D filter and shows that the filter has eliminated the undulations and so improved the accuracy of the solution.

In order to show the correct transfer between the surface energy and kinetic energy, a transient two-fluid oscillating bubble was simulated. The result was compared with the analytical expression for the frequency and the amplitude decay. This also showed reasonable agreement.

Rising bubbles immersed in a continuous phase are classical examples to corroborate any multi-fluids code. Bubbles rising with low and moderate Reynolds numbers were simulated and compared well with results in the literature [7,8]. The results showed small oscillations in the rising Reynolds number of the bubbles, occurring in coarse grids, and vanishing when the grid was subsequently refined. This effect is due to the fact that the bubble diameter is very small for the grid used, causing larger errors in the interfacial

tension distribution and was essentially eliminated when the grid was refined. A three-fluid rising bubble simulation was also designed to show the interaction between the fluids and the influence of the bouyancy forces on the flow. Bubble coalescences were simulated, with bubbles initially in-line and not in-line and the results were compared with numerical [6,22] and experimental [16] results of similar problems.

Finally, a two-fluid splashing drop was simulated to illustrate the robustness and applicability of the code to compute the interactions between free surfaces and interfaces. All these results together demonstrate that the new method can be used to simulate a number of problems and to investigate many physical phenomena involving incompressible multi-fluid flows that contain free surfaces.

Acknowledgements

We thank the financial support given by the Brazilian agencies FAPESP (Fundação de Amparo a Pesquisa do Estado de São Paulo), grant 01/12623-5 and 00/03385-0, and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), grant 460473/01-8.

References

[1] A.A. Amsden, F.H. Harlow, The SMAC method: a numerical technique for calculating incompressible fluid flows, Los Alamos Scientific Laboratory, Report LA-4370, 1970.
[2] V. Armenio, An improved MAC method (SIMAC) for unsteady high-Reynolds free surface flows, International Journal for Numerical Methods in Fluids 24 (1997) 185–214.
[3] J.U. Brackbill, D.B. Kothe, C. Zemach, A continuum method for modeling surface tension, Journal of Computational Physics 100 (1992) 335–354.
[4] A. Castelo, N. Mangiavacchi, J.A. Cuminato, A.O. Fortuna, J. Oliveira, M.F. Tomé, S. McKee, Surface tension implementation for GENSMAC2D code, in: Proceedings of the 15th Brazilian Congress of Mechanical Engineering (COBEM), Águas de Lindóia – SP, 1999 (CD-ROM).
[5] A. Castelo, M.F. Tomé, C.N.L. César, S. McKee, J.A. Cuminato, Freeflow: an integrated simulation system for three-dimensional free surface flows, Computing Visualization in Science 2 (2000) 199–210.
[6] L. Chen, Y. Li, A numerical method for two-phase flow with an interface, Environmental Modeling and Software 13 (1998) 247–255.
[7] A. Esmaeeli, G. Tryggvason, Direct numerical simulations of bubbly flows. Part 1. Low Reynolds number arrays, Journal of Fluid Mechanics 377 (1998) 313–345.
[8] A. Esmaeeli, G. Tryggvason, Direct numerical simulations of bubbly flows. Part 2. Moderate Reynolds number arrays, Journal of Fluid Mechanics 385 (1999) 325–358.
[9] V.G. Ferreira, M.F. Tomé, N. Mangiavacchi, A. Castelo, J.A. Cuminato, A.O. Fortuna, S. McKee, High order upwinding and the hydraulic jump, International Journal for Numerical Methods in Fluids 39 (2002) 549–583.
[10] J. Glimm, J. Grove, B. Lindquist, O. McBryan, G. Tryggvason, The bifurcation of tracked scalar waves, SIAM Journal on Scientific and Statistical Computing 9 (1988) 61–79.
[11] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Physics of Fluids 8 (1) (1965) 2182–2189.
[12] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, Journal of Computational Physics 39 (1981) 201–235.
[13] N. Mangiavacchi, A. Castelo, J.A. Cuminato, A.O. Fortuna, M.F. Tomé, L.G. Nonato, S. McKee, Numerical simulation of surface tension dominated axisymmetric free surface flows, in: Proceedings of ENCIT'2000, Porto Alegre – RS, 2000 (CD-ROM).
[14] M. Mäntyla, An Introduction to Solid Modeling, Computer Science Press, Rockville, MD, 1988.
[15] H. Miyata, Finite difference simulation of breaking waves, Journal of Computational Physics 65 (1986) 179–214.
[16] S. Narayanan, H.J. Groossens, N.W.F. Kossen, Coalescence of two bubbles rising in line at low Reynolds number, Chemical Engineering Science 29 (1974) 2071–2082.
[17] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, Journal of Computational Physics 79 (1988) 12–49.
[18] F.L.P. Santos, N. Mangiavacchi, A. Castelo, M.F. Tomé, V.G. Ferreira, J.A. Cuminato, Numerical simulation of multi-phase flows using the freeflow-2D system, in: Proceedings of 16th Brazilian Congress of Mechanical Engineering (COBEM), Uberlândia – MG, 2001 (CD-ROM).

[19] J.A. Sethian, Level Set Methods, Cambridge University Press, Cambridge, 1996.
[20] J.A. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, Cambridge, 1999.
[21] S. Shin, D. Juric, Modeling three-dimensional multiphase flow using a level contour reconstruction method for front-tracking without connectivity, Journal of Computational Physics 180 (2002) 427–470.
[22] F.S. Sousa, N. Mangiavacchi, A. Castelo, L.G. Nonato, M.F. Tomé, J.A. Cuminato, Simulation of 3D free-surface flows with surface tension, in: Proceedings of 16th Brazilian Congress of Mechanical Engineering (COBEM), Uberlândia – MG, 2001 (CD-ROM).
[23] F.S. Sousa, N. Mangiavacchi, A. Castelo, L.G. Nonato, M.F. Tomé, J.A. Cuminato, A mass conserving filter for the simulation of 3D free surface flows with surface tension, in: Proceedings of 22nd Iberian Latin–American Congress on Computational Methods in Engineering (CILAMCE), Campinas – SP, 2001 (CD-ROM).
[24] F.S. Sousa, N. Mangiavacchi, A. Castelo, L.G. Nonato, M.F. Tomé, J.A. Cuminato, Numerical simulations of 3D free-surface flows with surface tension, in: Proceedings 5th International Meeting on High Performance Computing for Computational Science (VECPAR), vol. 1, 2002, pp. 225–238.
[25] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions do incompressible two-phase flow, Journal of Computational Physics 114 (1994) 146–159.
[26] M.F. Tomé, S. McKee, GENSMAC: a comp. marker-and-cell method for free surface flows in general domains, Journal of Computational Physics 110 (1) (1994) 171–189.
[27] M.F. Tomé, A. Castelo, J. Murakami, J.A. Cuminato, R. Minghim, M.C.F. Oliveira, N. Mangiavacchi, S. McKee, Numerical simulation of axisymmetric free surface flows, Journal of Computational Physics 157 (2000) 441–472.
[28] M.F. Tomé, A. Castelo, J.A. Cuminato, N. Mangiavacchi, S. McKee, GENSMAC3D: a numerical method for solving unsteady three-dimensional free surface flows, International Journal for Numerical Methods in Fluids 37 (7) (2001) 747–796.
[29] G. Tryggvason, B. Bunner, O. Ebrat, W. Tauber, Computations of multiphase flows by a finite difference/front tracking method. I Multi-fluid flows, in: 29th Computational Fluid Dynamics, Lecture Series 1998-03, Von Karman Institute for Fluid Dynamics, 1998.
[30] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous incompressible multi-fluid flows, Journal of Computational Physics 100 (1991) 25–27.
[31] A.E.P. Veldman, M.E. Vogels, Axisymmetric liquid splashing under low gravity conditions, Acta Astronautica 11 (1984) 641–649.
[32] J.A. Viecelli, A computing method for incompressible flows bounded by moving walls, Journal of Computational Physics 65 (1971) 179–214.
[33] J.R. Welch, F.H. Harlow, J.P. Shannon, B.J. Daly, The MAC Method, Los Alamos Scientific Laboratory, Report LA-3425, 1965.